

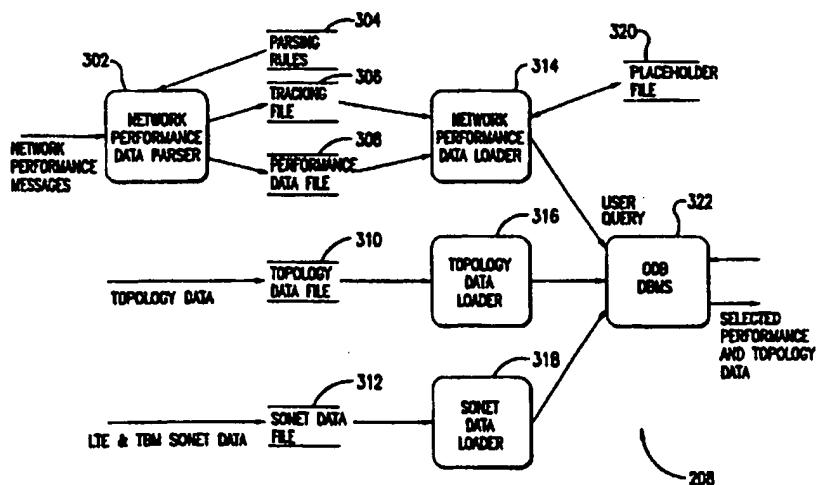


INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 17/30, H04J 1/16, H04M 1/24, 7/06	A1	(11) International Publication Number: WO 97/45801 (43) International Publication Date: 4 December 1997 (04.12.97)
---	----	---

(21) International Application Number: PCT/US97/09454 (22) International Filing Date: 28 May 1997 (28.05.97)	(81) Designated States: AU, CA, JP, MX, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).
(30) Priority Data: 08/655,153 28 May 1996 (28.05.96) US	Published <i>With international search report.</i>
(71) Applicant: MCI COMMUNICATIONS CORPORATION [US/US]; 1133 19th Street, N.W., Washington, DC 20036 (US).	
(72) Inventors: CUMMINS, Thomas, H.; 6841 S. Clermont Drive, Littleton, CO 80122 (US). FRY, Christopher, D.; 685 Paint Brush Lane, Monument, CO 80132 (US). HEGEMAN, Craig, A.; 4104 Summer Ridge Court, Apex, NC 27502 (US).	
(74) Agents: KESSLER, Edward, J. et al.; Sterne, Kessler, Goldstein & Fox P.L.L.C., Suite 600, 1100 New York Avenue, Washington, DC 20005-3934 (US).	

(54) Title: A SYSTEM AND METHOD FOR STORING AND RETRIEVING PERFORMANCE AND TOPOLOGY INFORMATION



(57) Abstract

A system and method for storing and retrieving performance and topology information of a telecommunications network. A network performance data parser (302) receives network performance data messages in real time, parses the messages according to parsing rules (304), reformats the messages according to the parsing rules, and writes out the reformatted messages to a performance data file (308). A network performance data loader (314) retrieves the reformatted messages from the performance data file and loads them into an open database management system (322). A topology data loader (316) retrieves weekly network topology data from a topology data file and loads it into the open database management system. A sonet data loader (318) retrieves daily network sonet data from a sonet data file (312) and loads it into the open database management system. The open database is easily accessible via any SQL interface.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LJ	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

A System and Method for Storing and Retrieving Performance and Topology Information

Background of the Invention

Field of the Invention

5 This invention relates generally to telecommunication networks, and more specifically, to provide trending analysis of a telecommunications network by monitoring network performance and topology information that spans days, weeks and months.

Related Art

10 A typical telecommunication network comprises multiple telecommunication switches located throughout a geographical area. When a customer makes a call, the call may be routed through one or more switches before reaching its destination. It is imperative that the network operate reliably and efficiently to ensure high quality network services and customer satisfaction.

15 The monitoring and maintenance of a telecommunications network is compounded by the fact that the number of network customers is continually increasing. Current network customers are using the network in new ways including access to Internet and the Wide World Web. Additionally, networks are continually expanding into new areas to reach new customers. Therefore, the volume of data transmitted over a telecommunications network is continually increasing along with the number of required monitoring points. A monitoring point is a location in the telecommunications network at which the network is being checked in terms of transmission errors, workload, and other performance statistics. As an example of the magnitude of this problem, there were over two (2) million monitoring points in the MCI telecommunications network at the end of 1995.

When monitoring a telecommunications network's performance, the network must be considered in its entirety. It is not sufficient to monitor a network in terms of individual segments because each segment of a network impacts the others. Therefore, monitoring network performance must include identifying trend analysis and root causes.

There are certain network conditions that can only be identified by trending analysis. For example, a bad Digital Cross Connect (DXC) 3/3 card may not be causing a problem that would show up on a real-time monitoring system. The bad card, however, would show up using long term trending analysis. Monitoring the trend of network performance over time identifies systemic problems in a particular manufacturer's product or indicates procedural problems in operating the network.

Contemporary network management systems do not provide complete or adequate network performance information. Typically, a management system provides a few report formats that contain limited performance information. The management system cannot provide trending analysis of network performance information that spans days, weeks and months. Additionally, contemporary systems do not combine network performance information with topology information. Therefore, network operators cannot correlate network performance with the placement of specific network devices.

The problem with contemporary network management systems is further evident when assisting in the testing of a customer's circuits prior to "turn-up." Turn-up occurs when a circuit has been provisional and has "live" data traffic. Without the availability of trending analysis, the testing of circuits is a manual intensive process. A system operator must monitor each circuit for several days. This procedure requires the operator to enter several circuit identifiers (as many as one hundred) each day for each circuit being monitored. After the monitoring is complete, the operator determines from the generated reports whether the circuit is ready to be "turned-up" for the customer.

5 Contemporary network management systems also collect and store performance data in a custom, internal format. That is, the systems do not store network information in an easily accessible standard format. By using a custom, internal format, the systems require custom application programs to be written to retrieve the performance data and present it in a meaningful format.

10 Therefore, there is a need for a telecommunications network monitoring system that provides both trending analysis and routine analysis. The monitoring system must support near-real-time, next day, and week-to-week analysis. Furthermore, both network performance and topology information must be incorporated into the same monitoring system. Therefore, the network performance information can be correlated with the topology information.

15 There is a further need for a telecommunications network monitoring system that provides an open and reliable collection point for network performance and topology information. Circuit availability and the percentage of error-free seconds can be monitored, thereby fulfilling tariff obligations and predicting problems prior to tariff impact.

20 There is still a further need for a telecommunications network monitoring system to provide a convenient and flexible means for the storage and retrieval of network performance and topology information. Easy and rapid access to such information assists network operators in monitoring the network in its entirety, as well as, in monitoring any specific monitoring point.

Summary of the Invention

The open database of the present invention solves the problems associated with contemporary network monitoring systems by providing a complete network monitoring system. The open database collects and stores network performance and topology information that spans days, weeks and months. Furthermore, the open database stores the network performance and topology information in a standard format. This allows easy access and retrieval of the information using commercial off-the-shelf software.

The open database receives three types of network information. First, network performance messages are sent in real time from multiple site controllers located throughout the telecommunications network. These messages are concentrated in a Single Local Support Element (LSE). The open database parses the performance messages as they are received from the LSE, performs any needed conversions, and translates the performance information into standard formatted records. The open database then loads the formatted records into the database management system (DBMS).

Second, the Light Terminating Equipment (LTE) and Transport Bandwidth Manager (TBM) devices send daily Synchronous Optical NETwork (SONET) messages to the DBMS. SONET messages provide performance information for the optical fiber transmissions on the public network. The open database loads these messages directly into the DBMS without the need for parsing or reformatting.

Third, the network Real Time Topology (RTT) system sends weekly topology messages to the open database. The topology messages provide information on the devices that comprise the network and on the configuration of the devices. The open database loads these messages directly into the DBMS without the need for parsing or reformatting.

The open database provides numerous advantages to network operators. Network operators have easy and rapid access to performance and topology information, thereby allowing the correlation of network performance to the

5

network topology. Operators can directly troubleshoot specific network devices. Examples of problems that can be found using the open database include: Digital System (DS) Circuit cable problems, multiplexer high speed card replacement, and lightwave transmitter replacement. Operators can also generate performance and time-of-day analyses for customer sites.

Additional features of this invention will become apparent from the following detailed description of the best mode for carrying out the invention and from appended claims.

Brief Description of the Figures

10

The present invention is described with reference to the accompanying drawings. In the drawings, like reference numbers indicate identical or functionally similar elements. Additionally, the left-most digit(s) of a reference number identifies the drawing in which the reference number first appears.

15

FIG. 1 is a block diagram of an exemplary computer system as used by the present invention;

FIG. 2 is a block diagram of an open database and the environment in which it operates;

FIG. 3 is a block diagram of the internal structure of the open database;

20

FIG. 4 is a control flow diagram representing the operation of a Network Performance Data Parser;

FIG. 5 is a control flow diagram representing the operation of Process Performance Message;

-6-

FIG. 6 is a control flow diagram representing the operation of Write Reformatted Performance Message to Performance Data File;

FIG. 7 is a control flow diagram representing the operation of Add Reformatted Message Data Records to Performance Data Files;

5 FIG. 8 is a control flow diagram representing the operation of Process Reload-Parsing-Rules Signal;

FIG. 9 is a control flow diagram representing the operation of a Network Performance Data Loader;

10 FIG. 10 is a control flow diagram representing the operation of Wait and Retrieve a Data Record from Performance Data File;

FIG. 11 is a control flow diagram representing the operation of Read a Data Record from the Performance Data File;

FIG. 12 is a control flow diagram representing the operation of Process Next Entry in Tracking File;

15 FIG. 13 is a control flow diagram representing the operation of a Topology Data Loader; and

FIG. 14 is a control flow diagram representing the operation of a Sonet Data Loader.

Detailed Description of the Preferred Embodiments

I. System Configuration

5 The preferred embodiment of the present invention is computer software executing within a computer system. FIG. 1 shows an exemplary computer system. The computer system 102 includes one or more processors, such as a processor 104. The processor 104 is connected to a communication bus 106.

10 The computer system 102 also includes a main memory 108, preferably random access memory (RAM), and a secondary memory 110. The secondary memory 110 includes, for example, a hard disk drive 112 and/or a removable storage drive 114, representing a floppy disk drive, a magnetic tape drive, a compact disk drive, a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as EPROM, or PROM), etc. which is read by and written to by a removable storage unit 116. 15 Removable storage unit 116, also called a program storage device or a computer program product, represents a floppy disk, magnetic tape, compact disk, etc. As will be appreciated, the removable storage unit 116 includes a computer usable storage medium having stored therein computer software and/or data. The removable storage drive 114 reads from and/or writes to a removable storage unit 116 in a well known manner.

20

25 The computer system 102 may also include other similar means for allowing computer programs or other instructions to be loaded. Such means can include, for example, a communications interface 118. Communications interface 118 allows software and data to be transferred between computer system 102 and external devices. Examples of communications interface 118 can include a modem, a network interface (such as an Ethernet card), a communications port, etc. Software and data transferred via communications interface 118 are in the form of signals which can be electronic, electromagnetic, optical or other signals capable of being received by communications interface 118.

5 In this document, the term "computer program product" is used to generally refer to removable storage unit 116, a hard disk installed in hard disk drive 112, and signals transferred via communications interface 118. These computer program products are means for providing software to a computer system 102.

10 In an embodiment where the invention is implemented using software, the software may be stored in main memory 108, or in a computer program product and loaded into computer system 102 using removable storage drive 114, hard disk drive 112, or communications interface 118. The software, when executed by the processor 104, causes the processor 104 to perform the functions of the invention as described herein.

15 In another embodiment, the invention is implemented primarily in hardware using, for example, a hardware state machine. Implementation of the hardware state machine so as to perform the functions described herein will be apparent to persons skilled in the relevant arts.

20 The preferred embodiment of the present invention is directed to a computer system 102 comprising a Digital Alpha Server 8400 executing Digital UNIX. Furthermore, the preferred embodiment of the Digital Alpha Server 8400 includes 100 gigabytes (GB) of disk space with 512 megabytes (MB) of RAM. It should be understood, however, that the invention is not limited to this embodiment. Instead, the invention is described with reference to this computer system for convenience purpose only. Other computer systems could alternatively be used.

25 Additionally, the preferred embodiment of the present invention is directed to software. Preferably, the software package Sybase System II (Sybase) executes on the preferred computer system. Sybase is a relational database product that is commercially available and is accessible via an SQL interface. Pertinent aspects of Sybase are described in many publicly available documents, such as Purba, S., *Developing Client/Server Systems Using Sybase SQL Server System XI*, John Wiley & Sons, ISBN 0471153389 (1996), which is incorporated herein by reference in its entirety. It should be understood that the invention is

described with reference to Sybase for convenience purpose only. Other software packages could alternatively be used.

2. *Data Flow for the Open Database*

5 FIG. 2 is a level 0 data flow diagram illustrating the external interfaces of the open database. A data flow diagram represents how data is used among the components of a computer software application. A level 0 data flow diagram illustrates the highest level of data interfaces between a software application and external sources. Therefore, in the level 0 data flow diagram of FIG. 2, the computer software application being depicted is the open database 208 which is
10 represented by the rounded-rectangle (or bubbangle). The LSE 202, PAVE 204, RTT 206, and ODB Clients 210 represent external sources of data that interface with the open database 208.

15 In a data flow diagram, the direction of an arrow connecting two objects represents the directional flow of data. For example, the Local Support Elements (LSE) 202 sends network performance messages to the open database 208. The PAVE 204 sends LTE and TBM sonet data to the open database 208. The RTT 206 sends topology data to the open database 208. The open database 208 receives and stores the network performance and topology information.

20 The ODB clients 210 have two data interfaces with the open database 208. First, ODB clients 210 send a user query to the open database 208 requesting for specific network performance and topology information. Second, the open database 208 returns to the ODB clients 210 the selected performance and topology data that matches the user query.

25 The single Local Support Element (LSE) 202 receives network performance information from multiple network site controller devices. Currently, there are approximately 200-300 site controllers located throughout the United States. The LSE 202 receives the performance information and sends it to the open database 208 in real time. There are ten (10) different types of network performance messages : generation 2 performance poll message -

ESFMU message, generation 2 performance poll message - CSU message, generation 2 performance nominal message, generation 2 performance poll message (crunched), generation 3 performance poll message (type 1 ESF circuit), Alcatel 1633 broadband DXC, DS3MU, E1MU, VC12, and SONET.

5 Example embodiments of the ten (10) network performance message formats discussed herein are further described in Table 1, below. Table 1 can be found in section 6 of the present specification and is self-explanatory. The embodiments of the network performance messages of the present invention comprise complete record formats. It would be apparent to one skilled in the
10 relevant art to develop alternative embodiments for network performance messages comprising a different number of fields and different field definitions.

15 The PAVE 204 receives Synchronous Optical NETwork (SONET) messages from Light Terminating Equipment (LTE) and Transport Bandwidth Manager (TBM). The SONET messages contain daily summaries pertaining to performance information on the optical fiber transmissions on the network. The
20 PAVE 204 receives the SONET messages and sends them to the open database 208 daily. There are nine (9) different types of SONET messages : pave_reports, sonet_ds3_reports, sonet_ds3_detail, sonet_sts1_reports, sonet_sts1_detail, sonet_oc12_reports, sonet_oc12_detail, sonet_oc48_reports, and
25 sonet_oc48_detail.

25 Example embodiments of the nine (9) SONET message formats discussed herein are further described in Table 2, below. Table 2 can be found in section 6 of the present specification and is self-explanatory. The embodiments of the network performance messages of the present invention comprise complete record formats. It would be apparent to one skilled in the relevant art to develop alternative embodiments for network performance messages comprising a different number of fields and different field definitions.

30 The Real Time Topology (RTT) system 206 receives daily updates from the topology database. The topology messages provide information on the various devices that comprise the network and on the configuration of those devices. The RTT 206 receives the topology information and sends it to the open

5 database 208 weekly. There are sixteen (16) different types of topology messages : ATT_EQUIP, CUSTOMER, CUSTOMER_CIRC, DS3_ROUTE, DXC31, DXC31_ALT, EQUIP_TYPES, FIBER_XREF, MUX, RADIO_DREI, RADIO_ROUTE, SEGMENT, STATION, STATION_NACS, SYS_ROUTE, and TRUNK.

10 Example embodiments of the sixteen (16) topology message formats discussed herein are further described in Table 3, below. Table 3 can be found in section 6 of the present specification and is self-explanatory. The embodiments of the network performance messages of the present invention comprise complete record formats. It would be apparent to one skilled in the relevant art to develop alternative embodiments for network performance 15 messages comprising a different number of fields and different field definitions.

15 The ODB Clients 210 send a user query to the open database 208 requesting for specific network performance and topology information. Because 20 the open database is implemented using a standard, relational database format, the ODB Clients 210 can access the performance and topology information using the structured query language (SQL). More specifically, the open database can be accessed and displayed using any commercial off-the-shelf software, such as Microsoft Excel, that implements SQL. Upon receiving a user query, the open 25 database 208 returns to the ODB clients 210 the selected performance and topology data that matches the user query.

25 FIG. 3 is a level 1 data flow diagram illustrating the components of the open database 210 and their relationship to the external interfaces (as shown in FIG. 2) and to each other. The components of the open database 210 include : the network performance data parser 302, network performance data loader 314, topology data loader 316, sonet data loader 318, and ODB DBMS 322.

30 Referring again to FIG. 2, the LSE 202 sends network performance data to the open database 208. More specifically, as shown in FIG. 3, the network performance data parser 302 establishes a TCP/IP connection with the external data source LSE 202 and acts as a server to the LSE 202. The network performance data parser 302 receives network performance messages from the

5 LSE 202 in real time, parses the messages, and generates reformatted performance messages according to the parsing rules 304. The reformatted performance messages are stored as data records in the performance data file 308, whereas information about the performance data file 308, such as filename and file size is stored in the tracking file 306.

10 Example embodiments of the parsing rules discussed herein are further described in Table 4, below. Table 4 can be found in section 6 of the present specification and is self-explanatory. The embodiments of the parsing rules of the present invention comprise complete record formats. It would be apparent to one skilled in the relevant art to develop alternative embodiments for the parsing rules comprising a different number of fields and different field definitions.

15 The network performance data parser 302 receives the network performance messages and parses them according to the parsing rules 304. The parsed messages are stored in the performance data file. There are nineteen (19) different types of parsed messages. The ESFMU messages include : monitor_ctrl, odb_sites, odb_equip_codes, esfmu_reports, esfmu_detail, esfmu_exceptions, and esfmu_cert_summary. The BBDXC messages include : bbdxc_reports, bbdxc_detail, and bbdxc_exceptions. The DS3MU messages include : ds3mu_reports, ds3mu_detail, and ds3mu_exceptions. The EIMU messages include : elmu_reports, elmu_detail, and elmu_exceptions. The OSCU messages include : ocsu_reports, oscu_detail, and oscu_exceptions.

20 Example embodiments of the nineteen (19) parsed message formats discussed herein are further described in Table 5. Table 5 can be found in section 6 of the present specification and is self-explanatory. The embodiments of the parsed messages of the present invention comprise complete record formats. It would be apparent to one skilled in the relevant art to develop alternative embodiments for parsed messages comprising a different number of fields and different field definitions.

25 The network performance data loader 314 reads the parsed messages from the performance data file 308 as they become available, according to the file information stored in the tracking file 306, and loads the parsed messages into the

5 ODB database management system (DBMS) 322. The network performance data loader 314 also uses a placeholder file 320 to store information on the current position in the tracking file 306 and performance data file 308 from which it is reading. The two-headed arrow between the network performance data loader 314 and the placeholder file 320 indicates that the network performance data loader 314 reads and writes data to the placeholder file 320.

Referring again to FIG. 2, the RTT 206 sends topology data to the open database 208. More specifically, as shown in FIG. 3, the RTT 206 writes the data directly into a topology data file 310. The topology data loader 316 reads the topology data from the topology data file 310 on a weekly basis and loads it directly into the ODB DBMS 322. Because the RTT 206 writes the topology data to the topology data file 310 in a format compatible with the ODB DBMS 322, the topology data loader 316 does not have to parse or reformat the data.

Referring again to FIG. 2, the PAVE 204 sends LTE & TBM sonet data to the open database 208. More specifically, as shown in FIG. 3, the PAVE 204 writes the data directly into a sonet data file 312. The sonet data loader 318 reads the sonet data from the sonet data file 312 on a daily basis and loads it directly into the ODB DBMS 322. Because the PAVE 204 writes the sonet data to the sonet data file 310 in a format compatible with the ODB DBMS 322, the sonet data loader 318 does not have to parse or reformat the data.

Referring again to FIG. 2, the ODB clients 210 send a user query to the open database 208. More specifically, as shown in FIG. 3, the ODB DBMS 322 receives the user query. The ODB DBMS 322 is a commercially available relational database that supports the SQL interfaces as well as specialized processing. The specialized processing includes : nightly database maintenance, user views, stored client queries, etc. Upon receiving a user query, the ODB DBMS 322 accesses its data tables containing the network performance and topology data to prepare a response. The ODB DBMS 322 returns to the ODB clients 210 the selected performance and topology data.

3. *Control Flow of the Network Performance Data Parser*

FIG. 4 is a control flow diagram representing the operation of the network performance data parser 302. The network performance data parser 302 begins executing at step 402 and immediately goes to step 404. In step 404, the network performance data parser 302 reads the parsing rules 304 from the parsing rules file. Continuing to step 406, the network performance data parser 302 determines whether the parsing rules file exists. If the parsing rules file does not exist, then the network performance data parser 302 proceeds to step 408 and exits.

Referring again to step 406, if it is determined that the parsing rules file exists, the network performance data parser 302 proceeds to step 410. In step 410, the network performance data parser 302 creates a pattern mapping, according to the parsing rules 304, for each of the network performance data messages that it will receive. Continuing to step 412, the network performance data parser 302 enters an infinite loop. The network performance data parser 302 loops forever waiting for the receipt of a network performance message.

In step 412, the network performance data parser 302 waits for an incoming network performance message. Once it receives a performance message, the network performance data parser 302 proceeds to step 414. In step 414, the network performance data parser 302 determines whether the received performance message is event data. Event data is a network performance message that is detailed in the parsing rules 304. If the received performance message is not event data, the network performance data parser 302 returns to step 412 and waits for the next network performance message to arrive.

Referring again to step 414, if the received performance message is event data, the network performance data parser 302 proceeds to step 416. In step 416, the network performance data parser 302 processes the received performance message. Step 416 is described in greater detail below. After processing the message, the network performance data parser 302 continues to step 418. In step 418, the network performance data parser 302 processes the reload-parsing-rules signal. The reload-parsing-rules signal indicates whether a new parsing rules file

is available for loading into the parsing rules 304. Step 418 is described in greater detail below. Continuing from step 418, the network performance data parser 302 returns to step 412 and waits for the next network performance message to arrive.

5 FIG. 5 is a control flow diagram of step 416 regarding the processing of a performance message. The network performance data parser 302 begins executing at step 502 and immediately goes to step 504. In step 504, the network performance data parser 302 matches the received performance message against the parsing rules 304. Continuing to step 506, the network performance data parser 302 determines whether a parsing rule has been found that matches the performance message. If a matching parsing rule has not been found, the network performance data parser 302 proceeds to step 508 and writes the received performance message to an error log. The network performance data parser 302 continues to step 516 and returns processing back to step 416 in FIG. 4.

10 15 Referring again to step 506, if the network performance data parser 302 determines that a matching parsing rule has been found, it proceeds to step 510. In step 510, the network performance data parser 302 parses the received performance message according to the matching parsing rule. Continuing to step 512, the network performance data parser 302 reformats the performance message into an ODB DBMS 322 reformatted message format. The network performance data parser 302 invokes one or more user functions to perform the reformatting of the performance data message.

20 25 After completing step 512, the network performance data parser 302 continues to step 514 where it writes the reformatted performance message to the performance data file 308. Step 514 is described in greater detail below. From step 514, the network performance data parser 302 continues to step 516 where it returns processing back to step 416 in FIG. 4.

30 FIG. 6 is a control flow diagram of step 514 regarding the writing of a reformatted performance message to the performance data file 308. The network performance data parser 302 begins executing at step 602 and immediately goes to step 604. In step 604, the network performance data parser 302 finds the

newest tracking file. The newest tracking file is the tracking file 306 with the most recent time/date stamp. Continuing to step 606, the network performance data parser 302 determines whether the tracking file 306 exists. If the tracking file 306 does not exist, the network performance data parser 302 proceeds to step 608. In step 608, the network performance data parser 302 creates a tracking file 306 before continuing to step 610. Referring again to step 606, if the network performance data parser 302 determines that the tracking file 306 does exist, it proceeds directly to step 610.

In step 610, the network performance data parser 302 finds the newest performance data file. The newest performance data file is the performance file 308 with the most recent time/date stamp. Continuing to step 612, the network performance data parser 302 determines whether the performance data file 308 exists. If the performance data file 308 does not exist, the network performance data parser 302 proceeds to step 614. In step 614, the network performance data parser 302 creates a performance data file 308 before continuing to step 616. Referring again to step 612, if the network performance data parser 302 determines that the performance data file 308 does exist, it proceeds directly to step 616.

In step 616, the network performance data parser 302 creates an entry in the tracking file 304 for referencing the selected performance data file 308. Continuing to step 618, the network performance data parser 302 adds the reformatted performance message data record to the performance data file 308. Step 618 is described in greater detail below. After step 618, the network performance data parser 302 continues to step 620 and returns processing to step 514 in FIG. 5.

FIG. 7 is a control flow diagram of step 618 regarding the adding of a reformatted performance message data record to the performance data file 308. The network performance data parser 302 begins executing at step 702 and immediately goes to step 704. In step 704, the network performance data parser 302 writes the reformatted data record to the performance data file 308. Continuing to step 706, the network performance data parser 302 determines

whether the performance data file 308 is full. If the performance data file 308 is not full, the network performance data parser 302 proceeds to step 712. Step 712 is described in greater detail below.

5 Referring again to step 706, if the network performance data parser 302 determines that the performance data file 308 is full, it proceeds to step 708. In step 708, the network performance data parser 302 closes the performance data file 308. Continuing to step 710, the network performance data parser 302 creates a new performance data file 308 and then goes to step 712.

10 In step 712, the network performance data parser 302 updates the performance data file entry in the tracking file 304 to account for the new data record. Continuing to step 714, the network performance data parser 302 creates a new entry in the tracking file 304. After step 714, the network performance data parser 302 continues to step 716 and returns processing to step 618 in FIG. 6.

15 FIG. 8 is a control flow diagram of step 418 regarding the processing the reload-parsing-rules signal. The network performance data parser 302 begins executing at step 802 and immediately goes to step 804. In step 804, the network performance data parser determines whether a reload-parsing-rules signal exists. If the reload-parsing-rules signal does not exist, the network performance data parser 302 proceeds to step 806 and returns processing to step 418 in FIG. 4.

20 Referring again to step 804, if the network performance data parser 302 determines that the reload-parsing-rules signal exists, it proceeds to step 808. In step 808, the network performance data parser 302 reads the parsing rules from the new parsing rules file. Continuing to step 810, the network performance data parser 302 creates the pattern mapping according to the new parsing rules. The network performance data parser 302 continues to step 812 to determine whether the pattern mapping failed. For example, the pattern mapping may fail due to syntax errors or file errors. If the network performance data parser 302 determines that the pattern mapping did not fail, then it knows that the parsing rules are valid and proceeds to step 806. In step 806, the network performance data parser 302 returns processing to step 418 in FIG. 4.

Referring again to step 812, if the network performance data parser 302 determines that the pattern mapping failed, it knows that the new parsing rules are invalid and proceeds to step 814. In step 814, the network performance data parser 302 reverts to the previous parsing rules 304. Continuing to step 806, the network performance data parser 302 returns processing to step 418 in FIG. 4.

4. *Control Flow of the Network Performance Data Loader*

FIG. 9 is a control flow diagram representing the operation of a network performance data loader 314. The network performance data loader 314 begins executing at step 902 and immediately goes to step 904. In step 904, the network performance data loader 314 connects to the ODB DBMS 322. Continuing to step 906, the network performance data loader 314 retrieves the list of specific monitoring points that indicates the monitoring points of the telecommunications network from which performance messages will be sent. From step 906, the network performance data loader 314 goes to step 908 at which the network performance data loader 314 enters an infinite loop. The network performance data loader 314 continues to loop while waiting for data records to be written to the performance data file 308.

In step 908, the network performance data loader 314 waits for the network performance data parser 302 to write a data record to the performance data file 308. When the network performance data parser 302 writes a record to the performance data file 308, the network performance data loader 314 retrieves the data record. Step 908 is described in more details below.

Continuing to step 910, the network performance data loader identifies the type of data record that it retrieved from the performance data file 308. After which, the network performance data loader 314 loads the data record into the corresponding table in the ODB DBMS 322. The network performance data loader 314 returns to step 908 to wait for the network performance data parser 302 to write another data record to the performance data file 308.

5 FIG. 10 is a control flow diagram representing the operation of step 908 and the process of a network performance data loader 314 waiting and retrieving a data record from the performance data file 308. The network performance data loader 314 begins executing at step 1002 and immediately goes to step 1004. In
10 step 1004, the network performance data loader 314 opens the oldest tracking file. The oldest tracking file is the tracking file 306 having the least recent time/date stamp. Continuing to step 1006, the network performance data loader 314 next opens the placeholder file 320. After step 1006, the network performance data loader 314 continues to step 1008.

15 10 In step 1008, the network performance data loader 314 reads the name of the performance data file 320 from which it will retrieve data records and the position in that performance data file 320 at which it will begin retrieving data records. Continuing to step 1010, the network performance data loader 314 goes to the position in the performance data file 320 according to the information it
20 15 read from the placeholder file 320 in step 1008. Once the network performance data loader 314 is at the correct position, the network performance data loader 314 continues to step 1012 and reads the data record from the performance data file 308 at that position. Step 1012 is described in greater detail below. Continuing to step 1014, the network performance data loader 314 returns processing to step 908 in FIG. 9.

25 FIG. 11 is a control flow diagram representing the operation of step 1012 and the process of a network performance data loader 314 reading a data record from the performance data file 308. The network performance data loader 314 begins executing at step 1102 and immediately goes to step 1104. In step 1104, the network performance data loader 314 reads the next data record from the performance data file 308 and updates its current position in the placeholder file 320. Continuing from step 1104, the network performance data loader 314 enters step 1106 to determine whether the performance data file 308 returned an end-of-file (EOF).

30 30 In step 1106, if the performance data file 308 does not return an EOF, the network performance data loader 314 proceeds to step 1108. In step 1108, the

-20-

network performance data loader 314 returns the data record that it had just read from the performance data file 308. Continuing to step 1110, the network performance data loader 314 returns the processing to step 1012 in FIG. 10.

Referring again to step 1106, if the performance data file 308 returns an
5 EOF, the network performance data loader 314 proceeds to step 1112. In step 1112, the network performance data loader 314 retrieves the next performance data file entry in the tracking file 306. Continuing to step 1114, the network performance data loader 314 determines whether the tracking file returned an EOF.

10 In step 1114, if the tracking file 306 returns an EOF, the network performance data loader 314 proceeds to step 1116. In step 1116, the network performance data loader 314 closes and deletes the current tracking file 308 because it has completed its processing of that tracking file 308. Continuing to step 1118, the network performance data loader 314 finds the oldest tracking file.
15 The oldest tracking file is the tracking file 306 with the least recent time/date stamp. The network performance data loader 314 continues to step 1120 where it determines whether the tracking file 306 exists.

20 In step 1120, if the tracking file 306 does not exist, the network performance data loader 314 proceeds to step 1110 because there are no more entries in the performance data file 308 to load into the ODB DBMS 322. In step 1110, the network performance data loader 314 returns processing to step 1012 in FIG. 10.

25 Referring again to step 1120, if the tracking file 306 does exist, the network performance data loader 314 proceeds to step 1122 because there are more entries in the performance data file 308 to process. In step 1122, the network performance data loader 314 processes the next entry in the tracking file 306. Step 1122 is described in greater detail below. Continuing to step 1108, the network performance data loader 314 returns the data record that it had just read from the performance data file 308 before continuing to step 1110. In step 1110, the network performance data loader 314 returns the processing to step 1012 in
30 FIG. 10.

-21-

5

Referring again to step 1114, if the tracking file 306 does not return an EOF, the network performance data loader 314 proceeds to step 1122. In step 1122, the network performance data loader 314 processes the next entry in the tracking file 306. Step 1122 is described in greater detail below. Continuing to step 1108, the network performance data loader 314 returns the data record that it had just read from the performance data file 308 before continuing to step 1110. In step 1110, the network performance data loader 314 returns the processing to step 1012 in FIG. 10.

10

15

20

25

FIG. 12 is a control flow diagram representing the operation of step 1122 and the processing of a next entry in the tracking file 306. The network performance data loader 314 begins executing at step 1202 and immediately goes to step 1204. In step 1204, the network performance data loader 314 returns the next performance data file entry in the tracking file 306. Continuing to step 1206, the network performance data loader 314 closes and deletes the performance data file 308. The network performance data loader 314 has completed the processing of all of the data records stored in the performance data file 308 which is indicated by the fact that step 1106 determined that the performance data file 308 returned an EOF. Therefore, the performance data file 308 is not needed and is closed and deleted in step 1206.

Continuing to step 1208, the network performance data loader 314 goes to the first position in the performance data file 308 identified in the performance data file entry in the tracking file 306. After step 1208, the network performance data loader continues to step 1210. In step 1210, the network performance data loader reads the next data record from the performance data file 308 and updates its position in the performance data file 308 in the placeholder file 320. Continuing to step 1212, the network performance data loader returns processing to step 1122 in FIG. 11.

5. *Control Flow of the Topology Data Loader*

FIG. 13 is a control flow diagram representing the operation of a topology data loader 316. The topology data loader 316 begins executing at step 1302 and immediately goes to step 1304. In step 1304, the topology data loader 316 connects to the ODB DBMS 322. Continuing to step 1306, the topology data loader 316 enters a loop to retrieve and process all of the topology data records stored in the topology data file 310.

5

10

In step 1306, the topology data loader 316 retrieves a data record from the topology data file 310. Continuing to step 1308, the topology data loader 316 determines whether the topology data file 310 returned an EOF. If the topology data file 310 returned an EOF, there are no more data records to process and load into the ODB DBMS 322. Therefore, the topology data loader 316 proceeds to step 1310 and exits.

15

20

Referring again to step 1308, if the topology data file 310 does not return an EOF, the topology data loader 316 proceeds to step 1312. In step 1312, the topology data loader 316 identifies the type of topology data record that it had retrieved from the topology data file 310. Continuing to step 1314, the topology data loader 316 loads the topology data record into the ODB DBMS 322. In the preferred embodiment, the topology data loader 316 uses the Sybase utility Bulk-Copy (BCP) to load the topology data record into the ODB DBMS 322. After the topology data record has been loaded, the topology data loader 316 returns to step 1306 to retrieve another data record from the topology data file 310.

6. *Control Flow of the Sonet Data Loader*

25

FIG. 14 is a control flow diagram representing the operation of a sonet data loader 318. The sonet data loader 318 begins executing at step 1402 and immediately goes to step 1404. In step 1404, the sonet data loader 318 connects to the ODB DBMS 322. Continuing to step 1406, the sonet data loader 318

enters a loop to retrieve and process all of the sonet data records stored in the sonet data file 312.

5 In step 1406, the sonet data loader 318 retrieves a data record from the sonet data file 312. Continuing to step 1408, the sonet data loader 318 determines whether the sonet data file 312 returned an EOF. If the sonet data file 312 returned an EOF, there are no more data records to process and load into the ODB DBMS 322. Therefore, the sonet data loader 318 proceeds to step 1410 and exits.

10 Referring again to step 1408, if the sonet data file 312 does not return an EOF, the sonet data loader 318 proceeds to step 1412. In step 1412, the sonet data loader 318 identifies the type of sonet data record that it had retrieved from the sonet data file 312. Continuing to step 1414, the sonet data loader 318 loads the sonet data record into the ODB DBMS 322. In the preferred embodiment, the sonet data loader 318 uses the Sybase utility Bulk-Copy (BCP) to load the sonet data record into the ODB DBMS 322. After the sonet data record has been loaded, the sonet data loader 318 returns to step 1406 to retrieve another data record from the sonet data file 312.

7. *Tables.*

The following Tables 1-5 are all self-explanitory as described above. As previously stated:

20 Table 1 comprises example embodiments of the ten (10) network performance message formats as discussed above with reference to FIGs. 2 and 3;

Table 2 comprises example embodiments of the nine (9) SONET message formats as discussed above with reference to FIGs. 2 and 3;

25 Table 3 comprises example embodiments of the sixteen (16) topology message formats as discussed above with reference to FIGs. 2 and 3;

Table 4 comprises example embodiments of the parsing rules as discussed above with reference to FIGs. 2 and 3; and

Table 5 comprises example embodiments of the nineteen (19) parsed message formats as discussed above with reference to FIGs. 2 and 3.

Generation 2 Performance Poll Message - ESFMU message

Format:

```
sequence_number`julian_day`device_identifier:EE`mm/dd/yy`hh:mm:ss`shelf,card`direction
type`res`les`rses`oof`fss`csu`ss`csu_est_err`csu_synch_es`csu_synch_fs`COMPL
```

Example:

```
888`293`CHUCN0: EE 10/20/95 00:45:00 17,28 E000000 000 900 000 900 000 900 COMPL
```

Note that in this example, there is no CSU data reported.

Explanation:

sequence_number	Sequence number, up to 7 digits. Reset at midnight.
julian_day	Julian date (day of the year), range of 1-366
device_identifier	Device Identifier. Contains: 3 character site id 3 character equipment identifier
EE	Indicates that this is a GEN 2 ESFMU performance message
mm/dd/yy	Date at the beginning of the reporting period.
hh:mm:ss	Time at the beginning of the reporting period.
shelf	Shelf identifier, numeric range of 01 through 18
card	Card identifier, numeric range of 01 through 28
direction	Direction (E=Network side, W=Customer Side of monitoring unit)
type	Type of data, 6 positions Position 1: 0=Non CSU data only 1=CSU data included 4 additional fields included ?=unrecognizable data included
res	Raw Errored Seconds
les	Link errored Seconds
rses	Raw Severely errored seconds
lses	Link severely errored seconds
oof	Out of Frame
fss	Failed Signal State
csu_ss	CSU current status byte. Values: UF00 00L0 The U bit is 1 if the F or L bit is 1 The F bit is 1 if unit in failed signal state The L bit is 1 if the unit is in DTE loopback state

TABLE 1

csu_esf_err	CSU ESF errors, numeric range of 0-65535
csu_synch_es	CSU synchronous errored seconds, numeric range of 0-900.
csu_synch_fs	CSU synchronous failed seconds, numeric range of 0-900.
COMPL	Indicates the end of the message.

Generation 2 Performance Poll Message - CSU message

Format:

```
sequence_number`julian_day`device_identifier: `EC`mm/dd/yy`hh:mm:ss`shelf,card`  
csu_ss`csu_esf_err`csu_synch_es`csu_synch_fs`COMPL
```

Example:

```
889`293`CHUCN0: EC 10/20/95 00:45:00 17,28 82 123 456 789 COMPL
```

Explanation:

sequence_number	Sequence number, up to 7 digits. Reset at midnight.
julian_day	Julian date (day of the year), range of 1-366
device_identifier	Device Identifier. Contains: 3 character site id 3 character equipment identifier
EC	Indicates that this is a GEN 2 CSU performance message
mm/dd/yy	Date at the beginning of the reporting period.
hh:mm:ss	Time at the beginning of the reporting period.
shelf	Shelf identifier, numeric range of 01 through 18
card	Card identifier, numeric range of 01 through 28
csu_ss	CSU current status byte. Values: UF00 00L0 The U bit is 1 if the F or L bit is 1 The F bit is 1 if unit is in failed signal state The L bit is 1 if the unit is in DTE loopback state
csu_esf_err	CSU ESF errors, numeric range of 0-65535
csu_synch_es	CSU synchronous errored seconds, numeric range of 0-900.
csu_synch_fs	CSU synchronous failed seconds, numeric range of 0-900.
COMPL	Indicates the end of the message.

Generation 2 Performance Poll Message (Crunched)

Format:

```
sequence_number`julian_day`device_identifier: `Edirection`mm/dd/yy`hh:mm:ss`shelf,card`
```

TABLE 1 (continued)

type`register_values,res,les,rses,lses,oof,fss,csu_ss,csu_est_err,csu_synch_es,
csu_synch_fs`COMPL

Examples:

1. 890 293 CHUCN0: EW 10/20/95 00:45:00 17,26 0 300053,900,900,900,900,,900
COMPL
2. 891 293 CHUCN0: EW 10/20/95 00:45:00 17,27 1 900,900,900,900,,900,12,34,56,78
COMPL
3. 892 293 CHUCN0: EW 10/20/95 00:45:00 17,28 ? 1,1,1 COMPL
4. 892 293 CHUCN0: EW 10/20/95 00:45:00 17,28 ? ,,,,? COMPL

Note that fields between two commas are assumed to be nominal. Fields absent at the end of the messages and before the "COMPL" are also nominal.

Explanation:

sequence_number	Sequence number, up to 7 digits. Reset at midnight.
julian_day	Julian date (day of the year), range of 1-366
device_identifier	Device Identifier. Contains: 3 character site id 3 character equipment identifier
E	Indicates that this is a GEN 2 ESFMU performance crunch message
direction	Direction (E=Network side, W=Customer Side or monitoring unit)
mm/dd/yy	Date at the beginning of the reporting period.
hh:mm:ss	Time at the beginning of the reporting period.
shelf	Shelf identifier, numeric range of 01 through 18
card	Card identifier, numeric range of 01 through 28
type	Type of crunch message. 0 - Indicates only ESFMU data is reported. 7 fields of data only. 1 - CSU data is included, 4 additional fields are available to report CSU data. If no data is present in the message then the data is inferred as nulls. ESFMU data is present. Generate the CSU report. ? - CSU data is encountered but is unrecognizable. ESFMU data is present. Generate the SCU report.

TABLE 1 (continued)

register_values	Register values, type 0 record only.
res	Raw Errored Seconds
les	Link errored Seconds
rses	Raw Severely errored seconds
lses	Link severely errored seconds
oof	Out of Frame
fss	Failed Signal State
csu_ss	CSU current status byte. Values: UF00 00L0 The U bit is 1 if the F or L bit is 1 The F bit is 1 if unit is in failed signal state The L bit is 1 if the unit is in DTE loopback state
csu_esf_err	CSU ESF errors, numeric range of 0-65535
csu_synch_es	CSU synchronous errored seconds, numeric range of 0-900.
csu_synch-fs	CSU synchronous failed seconds, numeric range of 0-900.
COMPL	Indicates the end of the message.

Generation 2 Performance Nominal Message

Format:

sequence_number`julian_day`device_identifier:PN`mm/dd/yy`hh:mm:ss`shelf`
XXXXXXXXXXXXXXXXXXXXXXXXXXXX`WXXXXXXXXXXXXXXXXXXXX`COMPL

Example:

871 293 ASTCN0: PN 10/20/95 00:45:00 01 E.2....?...--2?2.....W.2....?...--2?2.....
COMPL

Explanation:

sequence_number	Sequence number, up to 7 digits. Reset at midnight.
julian_day	Julian date (day of the year), range of 1-366
device_identifier	Device Identifier. Contains: 3 character site id 3 character equipment identifier

TABLE 1 (continued)

EC	Indicates that this is a GEN 2 CSU performance message
mm/dd/yy	Date at the beginning of the reporting period.
hh:mm:ss	Time at the beginning of the reporting period.
shelf	Shelf identifier, numeric range of 01 through 18
E	Direction - East side of the card reporting.
W	Direction - West side of the card reporting.
x	Each x represents a card number reporting data. There are 28 positions per direction.
Values:	
'.' - standard reporting has occurred. A separate ESFMU report should have been generated.	
'2' - Gen 2 card reporting nominal data but no CSU data was included.	
'C' - Gen 2 card reporting nominal data including CSU data also reporting nominal.	
'?' - Gen 2 card reporting nominal data but CSU data was not available.	
'.' - Card is off line. No data is reported for this period.	
'E' - Gen 3 ESFMU nominal	
'D' - Gen 3 D4 format nominal.	

COMPL Indicates the end of the message.

Generation 3 Performance Poll Message (type 1 ESF circuit)

Format:

sequence_number^julian_day^device_identifier:^15^mm/dd/yy^hh:mm:ss^shelf,card^1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28^COMPL

Example:

888 293 ASTCNO: 15 10/20/95 00:45:00 01,01 E1 1,2,3,,,6,7,8,9,10,
11,12,13,14,15,16,17,18,19

889 293 ASTCN0: 15 10/20/95 00:45:00 01,01 E1 -20 20.,22.23.24.25 COMPL

Explanation:

These messages can be broken into multiple lines. A continuation condition is indicated when a line ends with a "," and not the "COMPL" code. COntinuation lines will repeat the fixed information followed by a "-" character to indicate "Continuation". Any data field containing zeros (0) will be suppressed and not reported. The message wraps based on the length of the message and is completely dependent on the data reported by the cards.

TABLE 1 (continued)

sequence_number	Sequence number, up to 7 digits. Reset at midnight.
julian_day	Julian date (day of the year), range of 1-366
device_identifier	Device Identifier. Contains: 3 character site id 3 character equipment identifier
15	Indicates that this is a GEN 3 ESFMU performance message
direction	Direction (E=Network side, W=Customer Side of monitoring unit)
mm/dd/yy	Date at the beginning of the reporting period.
hh:mm:ss	Time at the beginning of the reporting period.
shelf	Shelf identifier, numeric range of 01 through 18
card	Card identifier, numeric range of 01 through 28
performance indicators	Each performance indicator, separated by a comma ",".
1- seconds in interval	
2- MMS	Maintenance Mode Seconds
3- NISS	Not-in-service seconds
4- ES-L	Errored seconds
5- SES-L	Severely Errored Seconds
6- LOSS-L	Loss of signal seconds
7- CV-L	Code Violation
8- SAS-P	SEF/AIS seconds
9- AISS-P	Alarm Indication Signal Seconds
10- CSS-P	Controlled Slip Seconds
11- ES-P	Errored Seconds
12- SES-P	Severely Error Seconds
13- UAS-P	Unavailable Seconds
14- CV-P	Code Violations
15- CV-PFE	Code Violations
16- ES-PFE	Errored Seconds
17- ESA-P	Errored Seconds
18- ESA-PFE	Errored Seconds
19- ESB-P	Errored Seconds
20- ESB-PFE	Errored Seconds
21- SES-PFE	Severely Errored Seconds
22- SEFS-PFE	Severely Errored Frame Seconds
23- CSS-PFE	Controlled Slip Seconds
24- UAS-PFE	Unavailable Seconds
25- ES-LFE	Errored Seconds
26- FEC-L	Failure Count
27- FEC-P	Failure Count
28- FEC-PFE	Failure Count

GEN3 performance messages take on a different look, due to the enhanced data collection capabilities of the GEN3 card. All data, including far-end CSU data is included in a single message. By default, all GEN3 performance messages are output in a zero-suppressed format.

TABLE 1 (continued)

Alcatel 1633 Broadband DXC

The Alcatel 1633 TL1 DXC reports the following information:

Format:

```
st1:P mm/dd/yy hh:mm:ss ppp berl perp evl evp fpe oofp esl esp sesl sesp uasp psc psd pes
fes aiss cpe loss COMPL
```

Explanation:

ppp	DXC port number
berl	Bit error Ratio Line count
perp	Bit Error Ratio Path count
cvl	Accumulated Coding Violation Line count
cvp	Accumulated Coding Violation Path count
fpe	Frame parity Error
oofp	Out of Frame path error
esl	Error Second Line count
esp	Error Second Path count
sesl	Severely Errored Seconds line count
sesp	Severely Errored Seconds path count
uasp	Unavailable Errored seconds Path
psc	Protection Switch counts
psd	Protection Switch Duration
pes	Parity Error Second Count
fes	Frame Error Second count
aiss	AIS Second count
cpe	C-bit parity error count
loss	Line Loss of signal Seconds

The equipment type (e.g. ST1) is converted to the equipment alpha indicator. ST1 is converted to "AA", ST2 is converted to "AB" and so on.

DS3MU**Format of Standard Message:**

```
site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx INF sii nis pb cb cf in
site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx LIN es loss
site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx PAR cv es esa esb ses COMPL*
site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx PAR - %efs bber uas %avl sef ais COMPL*
site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx CBT cv es esa esb ses
site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx CBT - %efs bber uas %avl
site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx FEP cv es esa esb ses
```

site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx FEP - %efs bber uas %avl sef ais
 site_idM3A: D mm/dd/yy hh:mm:ss ds3,mp xx All data is nominal COMPL

The COMPL flag is optional.

site_id

3 character site identifier

M3A:

Identifies this message as a DS3MU site controller message.

D

Type of message is 15 minute performance data, or is 24 hour performance data.

mm/dd/yy

Date that the message was generated, date at the beginning of the 15 minute interval.

hh:mm:ss

Time that the message was generated, time at the beginning of the 15 minute interval.

ds3

DS3MU shelf identifier.

mp

Monitor point number. There are 6 mp's per card, 90 mp's per shelf.

xx

Interval. '15' indicates 15 minute interval.

INF

General information.

LIN

Line level information.

PAR

Near-end parity information (M23).

CBT

C-bit parity information

FEP

Far-end (FEBE/FEAC) information.

DAT

TABLE 1 (continued)

TBD.

Near-End M23 application fields (cb="M23") :

xx	Command Exec	
sii	Seconds in Interval	
nis	Not in service seconds	0-900 for 15 min / 0-86400 for 24 hr
pb	Monitoring Mode	0-900 for 15 min / 0-86400 for 24 hr
cb	C-bit indicator	PB=Parity bit/FB-Frame Bit
cf	Change of format	CBT for c-bit/M23 for m23
in	Incomplete Period Indicator	FC=form.change/NC= no change
es	(ES-LINE) Errored Seconds	PI for incom/OK for complete
loss	(loss-line) Loss of Signal Seconds	0-900 for 15 min / 0-86400-for 24 hr
cv	(CV-PATH) Code Violation	0-900 for 15 min / 0-86400 for 24 hr
es	(ES-PATH) Errored Seconds	0-65535 for 15 min / 0-4194303 for 24 hr
esa	(ESA-PATH) Errored Seconds Type A	0-900 for 15 min / 0-86400 for 24 hr
esb	(ESB-PATH) Errored Seconds Type B	0-900 for 15 min / 0-86400 for 24 hr
ses	(SES-PATH) Severely Errored Seconds	0-900 for 15 min / 0-86400 for 24 hr
%efs	(%EFS-PATH) Present Err. Free Second	000.00% to 100.0%
bber	(BBER-PATH) Background Bit Error Rate	00-99 for mm/ -06 to -11 ee 15min/ -06 to -13 ee 24 hr
uas	(UAS-PATH) Unavailable Seconds	0-900 for 15 min / 0-86400 for 24 hr
%avL	(%AVL-PATH) Percent Err. Availability	000.00% to 100.0%
sef	(SEFS-PATH) Severely Err. Frame Seconds	0-900 for 15 min / 0-86400 for 14 hr
ais	(AISS-PATH) Alarm Indicat Signal Secs	0-900 for 15 min / 0-86400 for 24 hr

Near-end C-bit application fields (cb="CBT") :

cv	(CV-PATH) Code Violation	0-65535 for 15 min/ 0-4194303 for 24 hr
es	(ES-PATH) Errored Seconds	0-900 for 15 min / 0-86400 for 24 hr
esa	(ESA-PATH) Errored Seconds Type A	0-900 for 15 min / 0-86400 for 24 hr
esb	(ESB-PATH) Errored Seconds Type B	0-900 for 15 min / 0-86400 for 24 hr
ses	(SES-PATH) Severely Errored Seconds	0-900 for 15 min / 0-86400 for 24 hr
%efs	(%EFS-PATH) Percent Err. Free Second	000.00% to 100.0%
bber	(BBER-PATH) Background Bit Error Rate	00-99 for mm/ -06 to -11 ee 15 min/ -06 to -13 ee 24 hr
eas	(UAS-PATH) Unavailable Seconds	0-900 for 15 min / 0-86400 for 24 hr
%avL	(%AVL-PATH) Percent Err. Availability	000.00% to 100.0%

TABLE 1 (continued)

Far-end C-bit application fields (cb="CBT" and "FEP" line)

cv	(CV-PATH) Code Violation	0-65535 for 15 min / 0-4194303 for 24 hr
es	(ES-PATH) Errored Seconds	0-900 for 15 min / 0-86400 for 24 hr
esa	(ESA-PATH) Errored Seconds Type A	0-900 for 15 min / 0-86400 for 24 hr
esb	(ESB-PATH) Errored Seconds Type B	0-900 for 15 min / 0-86400 for 24 hr
ses	(SES-PATH) Severely Errored Seconds	0-900 for 15 min / 0-86400 for 24 hr
%efs	(%EFS-PATH) Percent Err. Free Second	000.00% to 100.0%
bber	(BBER-PATH) Background Bit Error Rate	00-99 for min / -06 to -11 ee 15 min/-06 to
uas	(UAS-PATH) Unavailable Seconds	-13 ee 24 hr
%avL	(%AVL-PATH) Percent Err. Availability	0-900 for 15 min / 0-86400 for 24 hr
sef	(SEFS-PATH) Severely Err. Frame	000.00% to 100.0%
ais	Seconds	0-900 for 15 min / 0-86400 for 24 hr
	(AISS-PATH) Alarm Indicat Signal Secs	0-900 for 15 min / 0-86400 for 24 hr

E1MU**1. Performance Poll Message**

Format (each performance indicator is zero-suppressed):

EIM: P hh:mm:ss nnn,c,mp
 aiiss,bberp,berp,besl,besp,besr,code,crc,esl,esp,esr,ess,fas,lofs,loss,prbs,rais,rebe,sesl,sesp,
 sesr,ses,usal,uasp,uasr,uass COMPL

Explanation:

TABLE 1 (continued)

hh:mm:ss	time at the beginning of the monitoring period
nnn	shelf name
c	card number (1-15)
mp	monitoring poing number (1-16)
aiss	Alarm Indication Signal Seconds count
bberp	Background Block Error Ratio Path
berp	Bit Error Ratio Path count
besl	Burst Errored Seconds Line count
besp	Burst Errored Seconds Path count
besr	Burst Errored Seconds Remote count
code	Code Error count
crc	Cyclic Redundancy Checksum error count
esl	Errored Seconds Line count
esp	Errored Seconds Path count
esr	Errored Seconds Remote count
ess	Errored Seconds Service count
fas	Frame Alignment Signal error count
lofs	Loss of Frame Seconds count
loss	Line Loss Of Signal Seconds
prbs	Pseudo Random Bit Sequence error count
rais	Remote Alarm Indication Seconds
rebe	Remote End Block Error count
sesl	Severely Errored Seconds Line count
sesp	Severely Errored Seconds Path count
sesr	Severely Errored Seconds Remote count
sess	Severely Errored Seconds Service count
uasl	Unavailable Seconds Line count
uasp	Unavailable Seconds Path count
uasr	Unavailable Seconds Remote count
uass	Unavailable Seconds Service count

2. Performance Nominal Message

The performance nominal or PN messages are sent on a card-by-card basis as a list of monitoring points.

Format:

EIM: PN mm:dd:yy:hh:mm:ss nnn,c,mp-mp,mp,mp,mp,mp,mp,mp COMPL

Explanation:

hh:mm:ss	time at the beginning of the monitoring period
nnn	shelf name
c	card number (1-15)

TABLE 1 (continued)

mp monitoring poing number (1-16)

Example:

Here is an example of a PN message where monitoring points 1 through 4, 7, 9 through 12, and 14 have reported zero data:

EIM: PN hh:mm:ss nnn,c 1-4,7,9-12,14 COMPL

OCSU/VC12

The Site Controller reports for the OCSU are similar to the Alcatel 1633SX and EIMU reports. Report line length is limited to the existing Site Controller internal message length of 128 bytes. The Performance Incomplete ("PI") and Performance Nominal ("PN") messages uses ranging where ever possible to reduce total message size. The Performance ("P") message will be issued for each port reporting valid non zero data in at least one field. The "P" messages will use zero suppression to reduce the total message size. This means that zeros will not be included in the data and parameters will be included up to the last non-zero parameter.

OC3 performance data from the OCSU includes individual validity flags for each parameter. If an OC3 "P" message is generated, any parameter flagged as invalid will be represented as "?".

Port numbering scheme used for these reports:

VC-12 ports

Ports 1-63	Facility side VC12s 1-63
Ports 64-126	Terminal side VC-12s 1-63

OC3 ports

Port 1	Facility side OC3
Port 2	Terminal side OC3

The 4 character data type field used for these reports:

3C12	Imbedded VC-12 data
OC3	OC3 data

Where " " represents a space.

"P" Message description:

OSU: P HH:MM:SS 3C12 aaa,p ES-V,SES-V,UAS-V,EB-V,ES-VFE,UAS-VFE,EB-VFE
COMPL

OSU: PN HH:MM:SS tttt aaa pl,p2-p3 COML

TABLE 1 (continued)

OSUL PI HH:MM:SS tttt aaa pl,p2-p3 COMPL

Where

HH:MM:SS	is the report start time
tttt	is the report MUX level
aaa	is the shelf name
p	is the port number
p1	is a single port
p2	is a port range starting port number
p3	is a port range ending port number
ES-V	is errored seconds
SES-V	is severely errored seconds
UAS-V	is unavailable seconds
EB-V	is errored blocks
ES-VFE	is errored seconds far end
SES-VFE	is severely errored seconds far end
UAS-VFE	is unavailable seconds far end
EB-VFE	is errored blocks far end
ESS	is section errored seconds
SESS	is section severely errored seconds
SEFS	is severely errored framing seconds
CVS	is section coding violations
CVP	is path coding violations

Example report:

OSU: P 02:15:00 3C12 FOO,1 „,45 COMPL

VC-12 facility side ports 2 through 7 are nominal

OSU: PN 02:15:00 3C12 FOO 2-7 COMPL

VC-12 facility side ports 8 through 63 and terminal side ports 1 through 63 are incomplete.

OSU: P1 02:15:00 3C12 FOO 8-126 COMPL

OC3 facility and terminal sides are nominal

OSU: PN 02:15:00 OC3 FOO 1-2 COMPL

SONET LTE and TBM

1. From the PAVE System

The following describes the SONET device performance information obtained via the PAVE system.

The file will be ASCII comma-delimited, with character fields quoted. Here is the file format:

TABLE 1 (continued)

Field	Field Name	Type	Width	Description
1	ODB_MUX	Character	3	Mux (3 Character Site code)
2	ODB_ALPHA	Character	6	Alpha System designator
3	ODB_NETYPE	Character	4	Device type (LTE or TBM)
4	ODB_LEVEL	Character	6	Service level (OC48, OC12 or DS3)
5	ODB_LINE	Character	3	NE line
6	ODB_DIR	Character	1	Direction (T or R)
7	ODB_DATE	Character	8	Date as YYYYMMDD
8	ODB_SPAN	Character	1	L (line), P (Path) or S (Sect)
9	ODB_PERCOL	Numeric	6	# periods collected (1 if collected OK)
10	ODB_PERMIS	Numeric	6	# periods missed (1 if not collected)
11	ODB_PERERR	Numeric	6	# error period (1 if errors in record)
12	ODB_ES	Numeric	6	ES count
13	ODB_SES	Numeric	6	SES count
14	ODB_SEFS	Numeric	6	SEFS count
15	ODB_UAS	Numeric	6	UAS count
16	ODB_AISS	Numeric	6	AISS count

2. SONET PM data via the LSE

The following information was obtained from Northern Telecom, S/DMS TransportNode OC-3/OC-12 NE-TBM, TL1 interface Description, Release 7 Standard (NT7E65DC 323-1111-190) March 1994.

The Report PM messages are reported every hour per NE. The message is used to automatically report the PM statistics from the NEs to the OS. Although the interval between REPT PM messages from a NE is guaranteed to be one hour, the actual time which the REPT PM message is issued depends on both the time which the TL1 connection was established and on the number of network elements in the system.

If the system contains N network elements, then the OS will not receive N REPT PM messages on the hour. Instead, the OS will receive N REPT PM messages spread out over the hour with an interval of $60/N$ minutes.

The network elements also report the hourly PM counts to the OPC spread out over the hour to prevent overloading the communications channel with PM data. Because of the two stage PM reporting, NE to OPC and OPC to TL1 interface, the REPT PM message will always report the PM counts for the hourly period two hours behind the actual hour to guarantee the PM counts are accurate. For example, the REPT PM message received at 12:15 will report the PM counts for the 10:00 period.

Only non-zero values are reported.

Format:

TABLE 1 (continued)

sequence_number^julian_day^REpT^PM^host_office^<source identifier>^<date>^
<time>crlf
A^a^REPT^PM^b crlf
^>"c:d,e,f,g,h,i,j,k"crlf
;

Where:

source identifier	NE name or OPC that originated the TL1 message
date	TL1 message origination date
time	TL1 message origination time
a	ATAG (automatic message tag)
b	SCCM (second command code modifier)
c	AID (access identifier)
d	MONTYPE (type of monitored parameter)
e	MONVAL (value of monitored parameter)
f	VLDTY (validity indicator) COMPL indicates complete
g	LOCN (alarm location) NEN=Near End
h	DIRN (alarm direction) AZ=transmit, ZA=receive
i	TMPER (accumulation time period) 1-HR=1 hour
j	MONDAT (date of beginning of PM period) MM-DD-YY
format	
k	MONTM (time of beginning of PM period) HH-MM format

Example:

12345319^REPT PM^OPC004MBLJ~~~~O0425^95-11-15^01:41:03
A^^34885 REPT PM T4X8
~~~1-12G1:LBCL,99,PRTL,NEND,AZ,1-HR,11-14,23-00"  
~~~1-12G2:LBCL,101,PRTL,NEND,AZ,1-HR,11-14,23-00"

a. The Access Identifier (AID)

This parameter contains the identification of the administrative view.

| SCCM | AID Value | Example |
|-----------|--|---------|
| OC48,T9X | <Eq_id>-<CP_name>-C<first chan no.>[-C<last chan no.>] | 1-A |
| OC12,T4X8 | <Eq_id>-<CP_name> | 1-12G1 |
| T3 | <Eq_id>-<CP_name>-<Port_id> | 1-3G1-3 |

Where:

TABLE 1 (continued)

Eq_id integer value from 1 to N (normally equals 1). N represents the variable nature of the upper limit of the parameter value.

CP_name 3G1 to 3G16, or 3P for a DS3 Mapper circuit pack
 12G1 to 12G8 for OC-12 interface circuit packs
 SG1 for STS-1 tributaries
 S3G1 for STS-1 path termination
 A, B for terminals and ADMs; AE, BW, AW, BE for regenerators on an OC-48 circuit pack
 ESIG1 or ESIG2 for an external synchronization interface circuit pack.
 SCP for shelf processor circuit pack
 MIU for maintenance interface circuit pack
 OPC for options controller circuit pack
 OW11, OW12 for an orderwire circuit pack
 OLAG1 to OLAG8 for optical line amplifiers
 HPTG1 to HPTG8 for high performance transmitters

first chan no 1 to 24. This is either the timeslot number for the STS-1 signal of the first STS-1 signal of the three making up an STS-3c (concatenated) signal.

last chan no 3, 6, 12, 15, 18, 21 or 24. This is the third of the three continuous timeslots in an sts-3c (concatenated) signal.

Port_id 1 to 3 for DS3 Mappers and STS-1 interfaces.

b. **DS3 Performance Monitoring**

The SCCM value for DS3 messages is "T3". All REPT PM TL1 messages have a LOCN of "NEND".

| Performance Measurement test | MONTYPE | DIRN | Sybase column name |
|------------------------------|---------|------|--------------------|
| Line Coding Violation | CVL | ZA | lcv |
| Line Error Second | ESL | ZA | les |
| Line Severely Errored Second | SESL | ZA | lses |
| Path Coding Violation | CVP | AZ | tpcv |
| | | ZA | pcv |
| Path Error Second | ESP | AZ | tpes |
| | | ZA | pes |
| Path Severely Errored Second | SESP | AZ | tpses |
| | | ZA | pses |

TABLE 1 (continued)

| | | | |
|------------------------------------|------|----|--------|
| Path Severely Errored Frame Second | OOF | AZ | tpsefs |
| | | ZA | psefs |
| Path AIS Seconds | AISS | AZ | tpaiss |
| | | ZA | paiss |
| Path Unavailable Seconds | UASP | AZ | tpuas |
| | | ZA | puas |
| Protection Switch Count | PSC | NA | psc |
| Protection Switch Duration | PSD | NA | psd |
| Laser bias current level | LBCL | NA | lbc1 |
| Laser power intensity transmitted | LPT | NA | lpt |

c. STS1 Performance Monitoring

The SCCM value for STS1 messages is "STS1". All REPT PM TL1 messages have a LOCN of "NEND".

| Performance Measurement test | MONTYPE | DIRN | Sybase column name |
|---------------------------------------|---------|------|--------------------|
| Line Coding Violation | CVL | ZA | lcv |
| Line Error Second | ESL | ZA | les |
| Line Severely Errored Second | SESL | ZA | lses |
| Line Unavailable Seconds | UASL | AZ | tluas |
| | | ZA | luas |
| Line Bipolar coding violation | BCVL | ZA | lbcv |
| Line Bipolar errored seconds | BESL | ZA | lbes |
| Line Bipolar severely errored seconds | BSESL | ZA | lbeses |
| Laser bias current level | LBCL | NA | lbc1 |
| Laser power intensity transmitted | LPT | NA | lpt |

TABLE 1 (continued)

d. OC12 Performance Monitoring

The SCCM value for OC12 messages is either "T4X8" or "OC12". All REPT PM TL1 messages have a LOCN of "NEND".

| Performance Measurement test | MONTYPE | DIRN | Sybase column name |
|---------------------------------------|---------|------|--------------------|
| Line Coding Violation | CVL | ZA | lcv |
| Line Error Second | ESL | ZA | les |
| Line Severely Errored Second | SESL | ZA | lses |
| Section Coding Violation | CVS | ZA | s_cv |
| Section Error Second | ESS | ZA | s_es |
| Section Severely Errored Second | SESS | ZA | s_ses |
| Section Severely Errored Frame Second | OOF | ZA | s_sefs |
| Protection Switch Count | PSC | NA | psc |
| Protection Switch Duration | PSD | NA | psd |
| Laser bias current level | LBCL | NA | lbcl |
| Laser power intensity transmitted | LPT | NA | lpt |

e. OC48 Performance Monitoring

The SCCM value for OC48 messages is either "T9X" or "OC48". All REPT PM TL1 messages have a LOCN of "NEND".

| Performance Measurement test | MONTYPE | DIRN | Sybase column name |
|---------------------------------------|---------|------|--------------------|
| Line Coding Violation | CVL | ZA | lcv |
| Line Error Second | ESL | ZA | les |
| Line Severely Errored Second | SESL | ZA | lses |
| Section Coding Violation | CVS | ZA | s_cv |
| Section Error Second | ESS | ZA | s_es |
| Section Severely Errored Second | SESS | ZA | s_ses |
| Section Severely Errored Frame Second | OOF | ZA | s_sefs |
| Protection Switch Count | PSC | NA | psc |

TABLE 1 (continued)

| | | | |
|-----------------------------------|------|----|-------|
| Protection Switch Duration | PSD | NA | psd |
| Laser bias current level | LBCL | NA | lblcl |
| Laser power intensity transmitted | LPT | NA | lpt |

f. Performance Nominal Date and Time Calculation

The `rpt_dt` (report date-time stamp) for the SONET performance nominal messages will be computed based upon the OPC date-time stamp and not that of the Network Element.

A single REPT PM message is received with no detail when all NEs within the OPC span of control are nominal. For example:

O04246 95-11-29 19:27:37
A 89266 REPT PM T3

;

The date-time stamp is for the OPC, not for the network element.

According to the TL1 interface description, the OPC retrieves PM data from each NE beginning 5 minutes after the hour. The actual time that the TL1 message is sent to SONM depends on the quantity of NEs in the span of control, the quantity of facilities provisioned, and the quantity of non-zero counts.

We have observed that the messages are received at least one hour after the time the OPCs poll the NEs.

The `rpt_dt` which will be stored in ODB will be based upon the OPC date-time stamp. 2 hours will be subtracted from OPC time and rounded down to the whole hour.

For instance, the above REPT PM message would be parsed and inserted into the ODB with a `aid` value of NULL, 3600 seconds in the interval, and a `rpt_dt` of "11/29/95 17:00".

TABLE 1 (continued)

SONET performance data is collected from two sources: PAVE and SONM. PAVE reports are sent to ODB once a day and include daily summaries. SONM data are sent in near-realtime via the LSE.

1. **pave_reports**

The **pave_reports** table is loaded "as-is" from the PAVE system.

| | | |
|----------------------|----------|--|
| site_code | char(6) | |
| p_equip_alpha | char(6) | Alpha system designator |
| p_netype | char(4) | Device type (LTE or TBM) |
| p_level | char(6) | Service level (OC48, OC12, or DS3) |
| p_line | char(3) | NE line |
| l_dir | char(1) | Direction (T or R) |
| p_date | char(8) | Date as YYYYMMDD |
| p_span | char(1) | Span. L=line, P=path, S=Section |
| ld_dt | datetime | Load date and time. The date and time the row was loaded into ODB. |
| p_percol | smallint | Number of periods collected (1 if collected OK) |
| p_permis | smallint | Number of periods missed (1 if not collected) |
| p_pererr | smallint | Number of error periods (1 if errors in record) |
| es | int | Errored seconds. |
| ses | int | Severely errored seconds. |
| sefs | int | SEES count |
| uas | int | Unavailable seconds |
| aiss | int | Alarm indication signal seconds |

2. **sonet_ds3_reports**

| | | |
|----------------|-------------|---|
| tid | char(6) | Source identifier in found in the TL1 message. |
| aid | varchar(20) | The access identifier of the network element reporting the PM data. The AID can be used to pinpoint the exact location on the shelf where the PM data is being sent from. |
| rpt_dt | datetime | The report date and time is obtained with the change of state message generated by the data acquisition process. |
| dtl_key | numeric(15) | Detail key. Points to a row in the sonet_ds3_detail table. |
| ld_dt | datetime | The date and time that the row was inserted into the table. |

TABLE 2

3. sonet_ds3_detail

| | | |
|---------|-------------|--|
| dtl_key | numeric(15) | |
| vldty | smallint | Validity indicator, 0=incomplete, 1=complete |
| sii | int | Seconds in interval |
| lcv | int | Line coding violation |
| les | int | Line error second |
| lses | int | Line severely errored second |
| tpcv | int | Transmit path code violation |
| pcv | int | Receive path code violation |
| tpes | int | Transmit path error second |
| pes | int | Receive path error second |
| tpses | int | Transmit path severely error second |
| pses | int | Receive path severely error second |
| tpsefs | int | Transmit path severely error frame second |
| psefs | int | Receive path severely error frame second |
| tpaiss | int | Transmit path ais seconds |
| paiss | int | Receive path ais seconds |
| tpuas | int | Transmit path unavailable seconds |
| puas | int | Receive path unavailable seconds |
| psc | int | Protection switch count |
| psd | int | Protection switch count |

4. sonet_sts1_reports

| | | |
|---------|-------------|---|
| tid | char(6) | Source identifier found in the TL1 message. |
| aid | varchar(20) | The access identifier of the network element reporting the PM data. The AID can be used to pinpoint the exact location on the shelf where the PM data is being sent from. |
| rpt_dt | datetime | The report date and time is obtained with the change of state message generated by the data acquisition process. |
| dtl_key | numeric(15) | Detail key. Points to a row in the sonet_sts1_detail table. |
| ld_dt | datetime | The date and time that the row was inserted into the table. |

TABLE 2 (continued)

5. sonet_sts1_detail

| | | |
|---------|-------------|---|
| dtl_key | numeric(15) | |
| vldty | smallint | Validity indicator, 0=incomplete, 1=complete |
| sii | int | Seconds in interval |
| lcv | int | Receive line coding violation |
| les | int | Receive line error seconds |
| lses | int | Receive line severely errored seconds |
| tluas | int | Transmit line unavailable seconds |
| luas | int | Receive line unavailable seconds |
| lbcv | int | Receive line bipolar coding violation |
| lbes | int | Receive line bipolar errored seconds |
| lbses | int | Receive line bipolar severely errored seconds |

6. sonet_oc12_reports

| | | |
|---------|-------------|---|
| tid | char(6) | Source identifier in found in the TL1 message. |
| aid | varchar(20) | The access identifier of the network element reporting the PM data. The AID can be used to pinpoint the exact location on the shelf where the PM data is being sent from. |
| rpt_dt | datetime | The report date and time is obtained with the change of state message generated by the data acquisition process. |
| dtl_key | numeric(15) | Detail key. Points to a row in the sonet_oc12_detail table. |
| ld_dt | datetime | The date and time that the row was inserted into the table. |

7. sonet_oc12_detail

| | | |
|---------|-------------|--|
| dtl_key | numeric(15) | |
| vldty | smallint | Validity indicator, 0=incomplete, 1=complete |
| sii | int | Seconds in interval |
| lcv | int | Line coding violation |
| les | int | Line error second |
| lses | int | Line severe error second |
| s_cv | int | Section coding violation |
| s_es | int | Section error second |
| s_ses | int | Section severely errored second |
| s_sefs | int | Section severely errored frame seconds |
| psc | int | Protection switch count |
| pds | int | Protecton switch duration |
| lblc | int | Laser bias current level |
| lpt | int | Laser power intensity transmitted |

TABLE 2 (continued)

8. sonet_oc48_reports

| | | |
|----------------|--------------------|--|
| <u>tid</u> | <u>char(6)</u> | Source identifier found in the TL1 message. |
| <u>aid</u> | <u>varchar(20)</u> | The access identifier of the network element reporting the PM data. |
| | | The AID can be used to pinpoint the exact location on the shelf where the PM data is being sent from. |
| <u>rpt_dt</u> | <u>datetime</u> | The report date and time is obtained with the change of state message generated by the data acquisition process. |
| <u>dtl_key</u> | <u>numeric(15)</u> | Detail key. Points to a row in the sonet_co12_detail table. |
| <u>ld_dt</u> | <u>datetime</u> | The date and time that the row was inserted into the table. |

9. sonet_oc48_detail

| | | |
|----------------|--------------------|--|
| <u>dtl_key</u> | <u>numeric(15)</u> | |
| <u>vldty</u> | <u>smallint</u> | Validity indicator, 0=incomplete, 1=complete |
| <u>ssi</u> | <u>int</u> | Seconds in interval |
| <u>lcv</u> | <u>int</u> | Line coding violation |
| <u>les</u> | <u>int</u> | Line error second |
| <u>lses</u> | <u>int</u> | Line severe error second |
| <u>s_cv</u> | <u>int</u> | Section coding violation |
| <u>s_es</u> | <u>int</u> | Section error second |
| <u>s_ses</u> | <u>int</u> | Section severely errored second |
| <u>s_sefs</u> | <u>int</u> | Section severely errored frame seconds |
| <u>psc</u> | <u>int</u> | Protection switch count |
| <u>pds</u> | <u>int</u> | Protecton switch duration |
| <u>lbcl</u> | <u>int</u> | Laser bias current level |
| <u>lpt</u> | <u>int</u> | Laser power intensity transmitted |

TABLE 2 (continued)

Column List for table: ATT-EQUIP

| | |
|-------------------------|----------|
| ATT_LTE_STN_CODE | int |
| ATT_LTE_STN_LINK | int |
| ATT_LTE_EQUIP_CODE | int |
| ATT_LTE_EQUIP_ALPHA | char(3) |
| ATT_LTE_MUX_LEVEL | char(4) |
| ATT_LTE_SLOT_NBR | int |
| ATT_LTE_SIDE | char(1) |
| ATT_LTE_TX_RX | char(1) |
| ATT_LTE_TRANS_TYPE | char(2) |
| ATT_DS3MU_EQUIP_CODE | int |
| ATT_DS3MU_EQUIP_ALPHA | char(3) |
| ATT_DS3MU_MUX_LEVEL | char(4) |
| ATT_DS3MU_SLOT_NBR | int |
| ATT_DS3MU_SIDE | char(1) |
| ATT_DS3MU_TX_RX | char(1) |
| ATT_DS3MU_MONITOR_FLAG | char(1) |
| ATT_DS3MU_CKT_TRUNK_TYP | char(12) |
| ATT_DS3MU_PARITY | char(1) |
| ATT_DS3MU_JUNC_TERM | int |

Column List for table: CUSTOMER

| | |
|--------------------|-------------|
| CUST_CIRC_ID | char(12) |
| TRUNK_ID | binary(8) |
| CUST_MULTI_PT_FLAG | char(1) |
| CUST_CODE | char(8) |
| CUST_NAME | varchar(30) |

Column List for table: CUSTOMER_CIRC

| | |
|--------------------|-----------|
| CUST_CIRC_ID8 | char(8) |
| TRUNK_ID | binary(8) |
| CUST_MULTI_PT_FLAG | char(1) |
| CUST_CODE | char(8) |
| CUST_NAME | char(30)) |

Column List for table: DS3_ROUTE

| | |
|-----------------|----------|
| DS3_STN_CD | int |
| DS3_EQUIP_CD | int |
| DS3_EQUIP_ALPHA | char(3) |
| DS3_CLOT_NBR | int |
| DS3_TRUNK_ID | int |
| DS3_SEQ_NBR | int |
| SEG_SEQ_NBR | int |
| DS3_DIR | char(1) |
| SYS_STRUNK_ID | int |
| SYS_STRUNK_REV | smallint |
| SYS_STRUNK_TYPE | char(1) |
| DS3_CONN_TYPE | char(1) |
| DS3_MON_PNT | smallint |

Column List for table: DXC31

| | |
|-----------------|----------|
| DXC31_DCM_EQUIP | char(10) |
| DXC31_DCM_ALPHA | char(3) |
| DXC31_DCM_PORT | int |
| DXC31_SITE | char(6) |
| DXC31_EQUIP | char(10) |
| DXC31_ALPHA | char(3) |
| DXC31_SLOT | int |

Column List for table: DXC31_ALT

| | |
|-----------------|----------|
| DXC31_DCM_EQUIP | char(10) |
| DXC31_DCM_ALPHA | char(3) |
| DXC31_DCM_PORT | int |
| DXC31_SITE | char(6) |
| DXC31_EQUIP | char(10) |
| DXC31_ALPHA | char(3) |
| DXC31_SLOT | int |

Column List for table: EQUIP-TYPES

| | |
|------------------|----------|
| EQUIP_CODE | int |
| EQUIP_ID | char(10) |
| EQUIP_FUNCTION | char(1) |
| EQUIP_MEDIA_TYPE | char(1) |
| EQUIP_ALIAS_I | char(10) |

TABLE 3 (continued)

| | |
|-----------------|----------|
| EQUIP_ALIAS_2 | char(10) |
| EQUIP_ALIAS_3 | char(10) |
| EQUIP_ALIAS_4 | char(10) |
| EQUIP_MUX_LVL_1 | char(4) |
| EQUIP_MUX_LVL_2 | char(4) |
| EQUIP_MUX_LVL_3 | char(4) |
| EQUIP_MUX_LVL_4 | char(4) |

Column List for table: FIBER_XREF

| | |
|-------------------------|---------|
| FIBER_SYS_STN1_CD | int |
| FIBER_SYS_STN2_CD | int |
| FIBER_SYS_ALPHA | char(3) |
| FIBER_SYS_EQUIP_CODE | int |
| FIBER_SYS_STATUS | char(1) |
| FIBER_SEQ_NBR | int |
| FIBER_FROM_STATION_CODE | int |
| FIBER_TO_STATION_CODE | int |
| FIBER_PR_ALPHA | char(3) |
| FIBER_TYPE_CNCT | char(1) |
| FIBER_PR_EQUIP_CODE | int |

Column List for table: MUX

(Note: The MUX table contains only channel capacity and does not relate to other topology tables.)

| | |
|----------------|---------|
| MUX_LEVEL | char(4) |
| MUX_CHANL_CAPY | int |

Column List for table: RADIO_DREI

| | |
|---------------------|---------|
| RADIO_DREI_STN1_CD | int |
| RADIO_DREI_STN2_CD | int |
| RADIO_DREI_ALPHA | char(3) |
| RADIO_DREI_EQUIP_CD | int |
| RADIO_SEQ_NBR | int |
| RADIO_SYS_STN1_CD | int |
| RADIO_SYS_STN2_CD | int |
| RADIO_SYS_ALPHA | char(3) |
| RADIO_SYS_EQUIP_CD | int |

Column List for table: RADIO_ROUTE

| | |
|--------------------|---------|
| RADIO_SYS_STN1_CD | int |
| RADIO_SYS_STN2_CD | int |
| RADIO_SYS_ALPHA | char(3) |
| RADIO_SYS_EQUIP_CD | int |
| RADIO_SEQ_NBR | int |
| RADIO_STN_CD | int |
| RADIO_TENTHS_MILES | int |

Column List for table: SEGMENT

| | |
|------------------------|-------------|
| SEG_TYPE | char(1) |
| SEG_STRUNK_DIR | smalling |
| SEG_STRUNK_ID | binary(8) |
| SEG_OCCUPY_TAS | varchar(36) |
| TRUNK_ID | binary(8) |
| SEG_SEQ_NBR | int |
| SEG_TYPE_CONNECT | char(1) |
| SEG_EQUIP_FUNCTION | char(1) |
| SEG_SLOT1_STATION_CODE | int |
| SEG_SLOT1_STATION_LINK | int |
| SEG_SLOT1_EQUIP_CODE | int |

TABLE 3 (continued)

| | |
|------------------------|-------------|
| SEG_SLOT1_EQUIP_ALPHA | char(3) |
| SEG_SLOT1_MUX_LEVEL | char(4) |
| SEG_SLOT1_SLOT_NBR | int |
| SEG_SLOT1_SLOT_SIDE | char(1) |
| SEG_SLOT1_SLOT_TX_RX | char(1) |
| SEG_SLOT2_STATION_CODE | int |
| SEG_SLOT2_STATION_LINK | int |
| SEG_SLOT2_EQUIP_CODE | int |
| SEG_SLOT2_EQUIP_ALPHA | char(3) |
| SEG_SLOT2_MUX_LEVEL | char(4) |
| SEG_SLOT2_SLOT_NBR | int |
| SEG_SLOT2_SLOT_SIDE | char(1) |
| SEG_SLOT2_SLOT_TX_RX | char(1) |
| SEG_MILEAGE | real |
| SEG_RESTORE | char(1) |
| SEG_RESTORE_ID | varchar(14) |

Column List for table: STATION

| | |
|-----------------------|-------------|
| STATION_CODE | int |
| STATION_ID | char(6) |
| STATION_NAME | varchar(32) |
| STATION_DIVSN | char(2) |
| STATION_STATUS | char(1) |
| STATION_TYPE | char(2) |
| STATION_ACTUAL_ENTITY | char(1) |
| STATION_LAT_SEC | int |
| STATION_LONG_SEC | int |

Column List for table: STATION-NACS

| | |
|-----------------------|-------------|
| STATION_NACS_ID | char(6) |
| STATION_CODE | int |
| STATION_ID | char(6) |
| STATION_NAME | varchar(32) |
| STATION_DIVSN | char(2) |
| STATION_STATUS | char(1) |
| STATION_TYPE | char(2) |
| STATION_ACTUAL_ENTITY | char(1) |
| STATION_LAT_SEC | int |
| STATION_LONG_SEC | int |

TABLE 3 (continued)

Column List for table: SYS_ROUTE

| | |
|------------------|---------|
| SYS_STN1_CD | int |
| SYS_STN2_CD | int |
| SYS_TECH | char(1) |
| SYS_ALPHA | char(3) |
| SYS_EQUIP_CD | int |
| SYS_SEQ_NBR | int |
| SYS_ROUTE_STN_CD | int |
| SYS_TRUNK_ID | int |

Column List for table: TRUNK

| | |
|------------------------|-----------|
| TRUNK_ALT_KEY_LEVEL | char(1) |
| TRUNK_ALT_KEY_STATUS | char(1) |
| TRUNK_BLOCK | char(4) |
| TRUNK_ID | binary(8) |
| TRUNK_PT1_STATION_CODE | int |
| TRUNK_PT1_STATION_LINK | int |
| TRUNK_PT1_EQUIP_CODE | int |
| TRUNK_PT1_EQUIP_ALPHA | char(3) |
| TRUNK_PT1_MUX_LEVEL | char(4) |
| TRUNK_PT1_SLOT_NBR | int |
| TRUNK_PT1_SLOT_SIDE | char(1) |
| TRUNK_PT1_SLOT_TX_RX | char(1) |
| TRUNK_PT1_HANDOFF | int |
| TRUNK_PT1_TAS_MUX | char(36) |
| TRUNK_PT2_STATION_CODE | int |
| TRUNK_PT2_STATION_LINK | int |
| TRUNK_PT2_EQUIP_CODE | int |
| TRUNK_PT2_EQUIP_ALPHA | char(3) |
| TRUNK_PT2_MUX_LEVEL | char(4) |
| TRUNK_PT2_SLOT_NBR | int |
| TRUNK_PT2_SLOT_SIDE | char(1) |
| TRUNK_PT2_SLOT_TX_RX | char(1) |
| TRUNK_PT2_HANDOFF | int |
| TRUNK_PT2_TAS_MUX | char(36) |
| TRUNK_SIZE | binary(8) |
| TRUNK_TYPE | char(1) |
| TRUNK_PATCH1_FLG | char(1) |
| TRUNK_PATCH2_FLG | char(1) |
| TRUNK ASGN | char(1) |
| TRUNK TECH | char(1) |
| TRUNK_LINE_CODE | char(1) |
| TRUNK_FRAME | char(1) |

TABLE 3 (continued)

| | |
|------------------|---------|
| TRUNK_END1_FRAME | char(1) |
| TRUNK_END2_FRAME | char(1) |
| TRUNK_SIGNAL | char(1) |
| TRUNK_FIBER_FLG | char(1) |
| TRUNK_SATELLITE | char(1) |
| TRUNK_PRIORITY | int |

TABLE 3 (continued)

```

RWCRegexp MONTH "[0-1][0-9]"
RWCRegexp DAY      "[0-3][0-9]"
RWCRegexp YEAR     "[0-9][0-9]"
RWCRegexp MDY      MONTH "/" DAY "/" YEAR
RWCRegexp YMD      YEAR "-" MONTH "-" DAY

RWCRegexp HOUR     "[0-2][0-9]"
RWCRegexp MIN_SEC  "[0-5][0-9]"
RWCRegexp HMS      HOUR ":" MIN_SEC ":" MIN_SEC
RWCRegexp MDY_HMS  MDY " " HMS
RWCRegexp YMD_HMS  YMD " " HMS

initialization {
    initialize();
}

event PERF_NOM {
    seqnum      "[0-9]+ {}"
    jul         "[0-9]+ {}"
    site_id    "... {}"
    msg_type   "CN"  {}
    eq_type    "[0-9]" {}
    rpt_type   ": PN" {}
    time_stamp MDY_HMS {}
    shelf      "[0-9][0-9]" {}
    comma      "," {}
    card        "[0-9][0-9]" {}
    east_data   " E[^]+ {}"
    west_data   " W[^]+ {}"
    eom        "COMPL[ \r\n]*"
    {
        store ("esfmu", catPipe("30",seqnum,jul,site_id,conv_shelf(eq_type, shelf),card,"87",
        time_stamp,cat(eq_type,shelf),east_data,west_data));
    }
}

event SONET {
    seqnum      "[0-9]+ {}"
    jul         "[0-9]+ {}"
    reptpm     "REPTPM" {}
    sitecodeid  "....." {}
    sp          "[]+" {}
    tid        "....." {}
}

```

TABLE 4

```

rpt_dt YMD_HMS {}
fields ".;" {}
eom "[^+]*"
{
sonet(seqnum,jul, sitecodeid,tid,rpt_dt, fields);
}
}

event GEN2CRUNCH0 {
seqnum "[0-9]+"
jul "[0-9]+"
site_id "..."
msg_type "CN"
eq_type "[0-9]"
rpt_type "; E"
dir "."
sp ""
time_stamp MDY_HMS {}
shelf "[0-9][0-9]"
comma ","
card "[0-9][0-9]"
how many "[0]"
fields "[0-9?]+"
eom "COMPL[ \r\n]*"
{
store ("esfmu", catPipe("40",seqnum,jul,site_id,
conv_shelf(eq_type,shelf),card,"87",time_stamp,cat(eq_type,shelf),dir,
fill-field-listO(fields,"7")));
}
}

event GEN2_typeques {
seqnum "[0-9]+"
jul "[0-9]+"
site_id "..."
msg_type "CN"
eq_type "[0-9]"
rpt_type "; E"
dir "."
sp ""
time_stamp MDY_HMS {}
shelf "[0-9][0-9]"
comma ","
card "[0-9][0-9]"
how_many "[?]"
fields "[0-9?]+"
eom "COMPL[ \r\n]*"
}

```

TABLE 4 (continued)

```

{
  store ("esfmu", catPipe("42",seqnum,jul,site_id,
    conv_shelf(eq_type,shelf),card,"87",time_stamp,cat(eq_type,shelf),dir,
    fill_field_listx(fields, "7")));
}
}

event GEN_type1 {
  seqnum      "[0-9]+ {}"
  jul         "[0-9]+ {}"
  site_id    "... {}"
  msg_type    "CN {}"
  eq_type     "[0-9]" {}
  rpt_type    ": E" {}
  dir         "." {}
  sp          " " {}
  time_stamp MDY_HMS {}
  shelf      "[0-9][0-9]" {}
  comma      "," {}
  card       "[0-9][0-9]" {}
  how_many   " 1" {}
  fields     "[0-9?]+ {}"
  eom        "COMPL[ \n]*"
{
  store ("esfmu", catPipe("41",seqnum,jul,site_id,
    conv_shelf(eq_type,shelf),card,"87",time_stamp,cat(eq_type,shelf),dir,
    fill_field_list1(fields, "11")));
}
}

event DXC_P {
  seqnum      "[0-9]+ {}"
  jul         "[0-9]+ {}"
  site_id    "... {}"
  msg_type    "ST" {}
  eq_type     "[0-9]" {}
  indicator   "; P" {}
  time_stamp MDY_HMS {}
  pppp      "[0-9]+ {}"
  berl      "[0-9]+ {}"
  perp      "[0-9]+ {}"
  cvl       "[0-9]+ {}"
  cvp       "[0-9]+ {}"
  fpe       "[0-9]+ {}"
  oofp      "[0-9]+ {}"
  esl       "[0-9]+ {}"
  esp       "[0-9]+ {}"
}

```

TABLE 4 (continued)

```

sesl    "[0-9]+ {}"
sesp    "[0-9]+ {}"
cses    "[0-9]+ {}"
uasp    "[0-9]+ {}"
psc     "[0-9]+ {}"
psd     "[0-9]+ {}"
pes     "[0-9]+ {}"
fes     "[0-9]+ {}"
aiss   "[0-9]+ {}"
fbe    "[0-9]+ {}"
cpe     "[0-9]+ {}"
loss   "[0-9]+ {}"
eom    "COMPL" {}
eom1   "[\r\n]*"
{
  store("dxc", catPipe("50", seqnum, jul, site_id, cat(matches("ST", msg_type, "A"),
    matches("1", eq_type, "A"), matches("2", eq_type, "B"), matches("3", eq_type, "C"),
    matches("4", eq_type, "D"), matches("5", eq_type, "E"),
    matches("6", eq_type, "F"), matches("7", eq_type, "G"), matches("8", eq_type, "H"),
    matches("9", eq_type, "I"), matches("10", eq_type, "J")), pppp, "140", time_stamp, eq_type,
    cat(adjust(cvl, "10", "0"), adjust(cvp, "10", "0"), adjust(fpc, "10", "0"),
    adjust(oofp, "10", "0"), adjust(esl, "10", "0"), adjust(esp, "10", "0"),
    adjust(sesl, "10", "0"), adjust(sesp, "10", "0"), adjust(cses, "10", "0"),
    adjust(uasp, "10", "0"), adjust(psc, "10", ""), adjust(psd, "10", "0"), adjust(pes, "10", "0"),
    adjust(fes, "10", "0"), adjust(aiss, "10", "0"), adjust(fbe, "10", "0"),
    adjust(cpe, "10", "0"), adjust(loss, "10", "0")));
}
}

event DXC-PN {
  seqnum    "[0-9]+ {}"
  jul       "[0-9]+ {}"
  site_id   "..." {}
  msg_type  "ST" {}
  eq_type   "[0-9]" {}
  indicator  ": P" {}
  time_stamp MDY_HMS {}
  fields   "[0-9-]+ {}"
  eom     "COMPL" {}
  eom1   "[\r\n]*"
{
  store ("dxc", catPipe("51", seqnum, jul, site_id,
    cat(matches("ST", msg_type, "A"), matches("1", eq_type, "A"),
    matches("2", eq_type, "B"), matches("3", eq_type, "C"),
    matches("4", eq_type, "D"), matches("5", eq_type, "E"), matches("6", eq_type, "F"),
    matches("7", eq_type, "G"), matches("8", eq_type, "H"), matches("9", eq_type, "I"),

```

TABLE 4 (continued)

```

        matches("10",eq_type,"J")), "0", "140", time_stamp, eq_type, fields));
    }
}

event DXC_PI {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "ST" {}
    eq_type     "[0-9]" {}
    indicator   ": PI" {}
    time_stamp  MDY_HMS {}
    fields      "[---0-9]+ {}"
    eom         "COMPL[ \r\n]*"
{
    store ("dxc",catPipe("52",seqnum,jul,site_id,
        cat(matches("ST",msg_type,"A"),matches("1",eq_type,"A"),matches("2",eq_type,"B"
        )),
        matches("3",eq_type,"C"),matches("4",eq_type,"D"),
        matches("5",eq_type,"E"),matches("6",eq_type,"F"),matches("7",eq_type,"G"),
        matches("8",eq_type,"H"),matches("9",eq_type,"I"),matches("10",eq_type,"J"),
        "0", "140", time_stamp, eq_type, fields));
}
}

event GEN315COMPL1 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "CN" {}
    eq_type     "[0-9]" {}
    rpt_type    ": 15" {}
    time_stamp  MDY_HMS {}
    shelf       "[0-9][0-9]" {}
    comma       "," {}
    card        "[0-9][0-9]" {}
    dir         "[EW]" {}
    typ         "1" {}
    fields      "[0-9?,-]+ {}"
    eom         " COMPL[ \r\n]*"
{
    store ("gen3",catPipe("60",seqnum,jul,site_id,
        conv_shelf(eq_type,shelf),card,"87",
        time_stamp,cat(eq_type,shelf),dir,gen3_compl(fields)));
}
}

```

TABLE 4 (continued)

```

event GEN315DASHCOMPL1 {
    seqnum      "[0-9] + " {}
    jul         "[0-9] + " {}
    site_id     "... " {}
    msg_type    "CN" {}
    eq_type     "[0-9]" {}
    rpt_type    ": 15" {}
    time_stamp  MDY_HMS {}
    shelf       "[0-9][0-9]" {}
    comma       "," {}
    card        "[0-9][0-9]" {}
    dir         "[EW]" {}
    type        "[1]" {}
    dash        "- " {}
    strt        "[0-9]+ " {}
    sp          " " {}
    fields      "[0-9,\-]+ " {}
    com         " COMPL[\r\n]*"
    {
        store ("gen3",catPipe("60",seqnum,jul,site_id,
            conv_shelf(eq_type,shelf),card,"87",
            time_stamp,cat(eq_type,shelf),dir,gen3_incom("DC",seqnum,jul,site_id,
            msg_type,eq_type,rpt_type,time_stamp,shelf,card,dir,type,fields)));
    }
}

event GEN315INCOMNODASHNOCOMPLI {
    seqnum      "[0-9] + " {}
    jul         "[0-9] + " {}
    site_id     "... " {}
    msg_type    "CN" {}
    eq_type     "[0-9]" {}
    rpt_type    ": 15" {}
    time_stamp  MDY_HMS {}
    shelf       "[0-9][0-9]" {}
    comma       "," {}
    card        "[0-9][0-9]" {}
    dir         "[EW]" {}
    type        "1" {}
    sp          " " {}
    fields      "[0-9,\-]+ " {}
    com         "[\r\n]*"
    {
        gen3_incom(" NN",seqnum,jul,site_id,msg_type,eq_type,rpt_type,time_stamp,
            shelf,card,dir,type,fields);
    }
}

```

TABLE 4 (continued)

```

event GEN315INCOMDASHNOCOMPL1 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id    "... {}"
    msg_type   "CN" {}
    eq_type    "[0-9]" {}
    rpt_type   ": 15" {}
    time_stamp MDY_HMS {}
    shelf      "[0-9{2}" {}
    card       "[0-9]{2}" {}
    dir        "[EW]" {}
    type       "[1]" {}
    dash      " - " {}
    strt      "[0-9]+"
    sp         " " {}
    fields    "[0-9\,-]+"
    com        "[\r\n]*"
    {
        gen3_incom("DN",seqnum,jul,site_id,msg_type,eq_type,rpt_type,time stamp,
                   shelf,card,dir,type,fields);
    }
}

event GEN315INCOMDASHNOCOMPL0 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id    "... {}"
    msg_type   "CN" {}
    eq_type    "[0-9]" {}
    rpt_type   ": 15" {}
    time_stamp MDY_HMS {}
    shelf      "[0-9][0-9]" {}
    comma     ","
    card       "[0-9][0-9]" {}
    dir        "[EW]" {}
    type       "[0]" {}
    dash      " - " {}
    strt      "[0-9]+"
    sp         " " {}
    fields    "[0-9\,-]+"
    com        "[\r\n]*"
    {
    }
}

event E1MU {
    seqnum      "[0-9] + {}"

```

TABLE 4 (continued)

```
    jul      "[0-9] + {}"
    site_id  "..." {}
    msg_type "EIM" {}
    rpt_type  "; P" {}
    time_stamp MDY_HMS {}
    shelf    "..." {}
    comma    "," {}
    card     "[0-9] + {}"
    comma1   "," {}
    mp      "[0-9]" {}
    fields  "[0-9?,-] + {}"
    eom     " COMPL" {}
    eom1    "[\r\n]*"
    {
        store ("dxc",catPipe("80",seqnum,jul,site_id,shelf,card,"153",
            time_stamp,mp,gen3_compl(fields)));
    }
}

event E1MUPI {
    seqnum   "[0-9] + {}"
    jul      "[0-9] + {}"
    site_id  "..." {}
    msg_type "EIM" {}
    rpt_type  "; PI" {}
    time_stamp MDY_HMS {}
    shelf    "..." {}
    comma    "," {}
    card     "[0-9] + {}"
    fields  "[0-9,-] + {}"
    eom     " COMPL[\r\n]*"
    {
        store ("dxc",catPipe("82",seqnum,jul,site_id,shelf,card,"153",time_stamp,fields));
    }
}
```

TABLE 4 (continued)

```

event E1MUPN {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "E1M" {}
    rpt_type    "; PN" {}
    time_stamp  MDY_HMS {}
    shelf       "... {}"
    comma       ";" {}
    card        "[0-9] + {}"
    fields      "[0-9?,-,] + {}"
    eom         " COMPL[\r\n]*"
    {
        store ("dxc",catPipe("81",seqnum,jul,site_id,shelf,card,time_stamp,fields));
    }
}

event E1MU2 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "E1M" {}
    rpt_type    "; P" {}
    time_stamp  MDY_HMS {}
    shelf       "... {}"
    comma       ";" {}
    card        "[0-9] + {}"
    commal      ";" {}
    mp          "[0-9] + {}"
    fields      "[0-9?,-,] + {}"
    eom         " COMPL" {}
    eom1        "[\r\n]*"
    {
        store ("dxc",catpipe("80",seqnum,jul,site_id,shelf,card,"153",
                               time_stamp,mp,gen3_compl(fields)));
    }
}

event E1MUPI2 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "E1M" {}
    rpt_type    "; PI" {}
    time_stamp  MDY_HMS {}
    shelf       "... {}"
    comma       ";" {}
}

```

TABLE 4 (continued)

```

card  "[0-9]+{}"
fields "[0-9,,-]+{}"
eom   "COMPL[\r\n]*"
{
  store ("dxc",catPipe("82",seqnum,jul,site_id,shelf,card,time_stamp,fields));
}
}

event E1MUPN2 {
  seqnum      "[0-9] + {}"
  jul         "[0-9] + {}"
  site_id     "...{}"
  msg_type    "E1M" {}
  rpt_type    ": PN" {}
  time_stamp  MDY_HMS {}
  shelf       "...{}"
  comma       "," {}
  card        "[0-9]+{}"
  fields      "[0-9,,-]+{}"
  eom         "COMPL[\r\n]*"
{
  store ("dxc",catPipe("81",seqnum,jul,site_id,shelf,card,time_stamp,fields));
}
}

event DS3MUALLNOM15 {
  seqnum      "[0-9] + {}"
  jul         "[0-9] + {}"
  site_id     "...{}"
  msg_type    "M3A" {}
  rpt_type    ": D" {}
  time_stamp  MDY_HMS {}
  spl         " " {}
  shelf       "...{}"
  comma       "," {}
  card        "[0-9]+{}"
  type        " 15" {}
  inf         " All" {}
  fields      "data is nominal" {}
  eom         " COMPL[\r\n]*"
{
  store ("dxc",catPipe("70",seqnum,jul,site_id,shelf,card,"115",time_stamp,"0"));
}
}

event DS3MUALLNOM24 {
  seqnum      "[0-9] + {}"

```

TABLE 4 (continued)

```

    jul      "[0-9] + {}"
    site_id  "... {}"
    msg_type "M3A" {}
    rpt_type ": 24" {}
    time_stamp MDY_HMS {}
    spl      " " {}
    shelf   "... {}"
    comma   "," {}
    card    "[0-9]+{}"
    type    " 24" {}
    inf     " All" {}
    fields  "data is nominal" {}
    eom     " COMPL[\r\n]*"
    {
    }
}

event DS3M24dat {
    seqnum  "[0-9] + {}"
    jul     "[0-9] + {}"
    site_id "... {}"
    msg_type "M3A" {}
    rpt_type ": 24" {}
    time_stamp MDY_HMS {}
    spl      " " {}
    shelf   "... {}"
    comma   "," {}
    card    "[0-9]+{}"
    type    " 24" {}
    inf     " DAT" {}
    dt      "[0-9/]+{}"
    eom     "[\r\n]*"
    {
    }
}

event DS3MUALLNOM24 {
    seqnum  "[0-9] + {}"
    jul     "[0-9] + {}"
    site_id "... {}"
    msg_type "M3A" {}
    rpt_type ": 24" {}
    time_stamp MDY_HMS {}
    spl      " " {}
    shelf   "... {}"
    comma   "," {}
    card    "[0-9]+{}"
}

```

TABLE 4 (continued)

```

type      " 24" {}
inf      " All" {}
fields   "data is nominal" {}
eom      " COMPL[\r\n]*"
{
}
}

event DS3MUIINF {
    seqnum      "[0-9] + " {}
    jul         "[0-9] + " {}
    site_id     "... " {}
    msg_type    "M3A" {}
    rpt_type    ": D" {}
    time_stamp  MDY_HMS {}
    spl         " " {}
    shelf       "... " {}
    comma       "," {}
    card         "[0-9]+ " {}
    type         " 15" {}
    inf         " INF" {}
    fields      "[0-9?,A-Z]+ " {}
    eom         "[\r\n]*"
{
    ds3(seqnum,jul,site_id,msg_type,rpt_type,time_stamp,spl,shelf,card,type, inf,fields);
}
}

event DS3MUIINF24 {
    seqnum      "[0-9] + " {}
    jul         "[0-9] + " {}
    site_id     "... " {}
    msg_type    "M3A" {}
    rpt_type    ": D" {}
    time_stamp  MDY_HMS {}
    spl         " " {}
    shelf       "... " {}
    comma       "," {}
    card         "[0-9]+ " {}
    type         " 15" {}
    inf         " INF" {}
    fields      "[0-9?,A-Z]+ " {}
    eom         "[\r\n]*"
{
}
}

```

TABLE 4 (continued)

```

event DS3MULIN {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "..." {}
    msg_type    "M3A" {}
    rpt_type    ": D" {}
    time_stamp  MDY_HMS {}
    spl         " " {}
    shelf       "..." {}
    comma       ";" {}
    card         "[0-9]+{}"
    type         " 15" {}
    lin          " LIN" {}
    fields      "[0-9?,Z-Z]+{}"
    eom          "[\r\n]*"
    {
        ds3(seqnum,jul,site_id,msg_type,rpt_type,time_stamp,spl,shelf,card,type,lin,fields);
    }
}

event DS3MULIN24 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "..." {}
    msg_type    "M3A" {}
    rpt_type    ": 24" {}
    time_stamp  MDY_HMS {}
    spl         " " {}
    shelf       "..." {}
    comma       ";" {}
    card         "[0-9]+{}"
    type         " 24" {}
    lin          " LIN" {}
    fields      "[0-9?,Z-Z]+{}"
    eom          "[\r\n]*"
    {
    }
}

event DS3MUPAR1 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "..." {}
    msg_type    "M3A" {}
    rpt_type    ": D" {}
    time_stamp  MDY_HMS {}
    spl         " " {}
}

```

TABLE 4 (continued)

```

shelf      "... {}"
comma     "," {}
card      "[0-9]+" {}
type      "15" {}
par      " PAR" {}
fields   "[0-9]+" {}
eom      "[\r\n]*"
{
ds3(seqnum,jul,site_id,msg_type,rpt_type,time_stamp,spl,shelf,card,type, par,fields);
}
}

event DS3MUPAR124 {
seqnum   "[0-9] + {}"
jul      "[0-9] + {}"
site_id  "... {}"
msg_type "M3A" {}
rpt_type ": D" {}
time_stamp MDY_HMS {}
spl      " " {}
shelf    "... {}"
comma   "," {}
card      "[0-9]+" {}
type      " 24" {}
par      " PAR" {}
fields   "[0-9]+" {}
eom      "[\r\n]*"
{
}
}

event DS3MUPARCOMPL {
seqnum   "[0-9] + {}"
jul      "[0-9] + {}"
site_id  "... {}"
msg_type "M3A" {}
rpt_type ": D" {}
time_stamp MDY_HMS {}
spl      " " {}
shelf    "... {}"
comma   "," {}
card      "[0-9]+" {}
type      " 15" {}
inf      " PAR" {}
dash    " - "
}

```

TABLE 4 (continued)

```

fields      "[0-9 %.\-]+{}"
eom1      "COMPL"
eom      "[\r\n]*"
{
store ("dxc",catPipe("70",seqnum,jul,site_id,shelf,card,"115"
    ,time_stamp,ds3(seqnum,jul,site_id,msg_type,rpt_type,time_stamp,sp1,
    shelf,card,type,"PDC",fields)));
}
}

event DS3MUPARDashnoCOMPL {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "M3A"
    rpt_type    ": D"
    time_stamp  MDY_HMS {}
    sp1         " " {}
    shelf       "... {}"
    comma       ","
    card         "[0-9]+{}"
    type         " 15"
    inf          " PAR"
    dash         "- "
    fields      "[0-9 %.\-]+{}"
    eom         "[\r\n]*"
{
    ds3 (seqnum,jul,site_id,msg_type,rpt_type,time_stamp,sp1,shelf,card,type,
    "PDN",fields);
}
}

event DS3MUFEPCOMPL {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "M3A"
    rpt_type    ": D"
    time_stamp  MDY_HMS {}
    sp1         " " {}
    shelf       "... {}"
    comma       ","
    card         "[0-9]+{}"
    type         " 15"
    inf          " FEP"
    dash         "- "
    fields      "[0-9 %.\-]+{}"
}

```

TABLE 4 (continued)

```

eom1      "[M]*COMPL" {}
eom      "[\r\n]*"
{
  store ("dxc",catPipe("70",seqnum,jul,site_id,shelf,card,"115"
    ,time_stamp,ds3(seqnum,jul,site_id,msg_type,rpt_type,time_stamp,sp1,
    shelf,card,type,"FDC",fields)));
}
}

event DS3MUFEP1 {
  seqnum      "[0-9] + {}"
  jul         "[0-9] + {}"
  site_id     "... {}"
  msg_type    "M3A" {}
  rpt_type    ": D" {}
  time_stamp  MDY_HMS {}
  sp1         " " {}
  shelf       "... {}"
  comma       "," {}
  card         "[0-9]+ {}"
  type         " 15" {}
  par          " FEP" {}
  fields      "[0-9]+ {}"
  eom         "[\r\n]*"
  {
    ds3 (seqnum,jul,site_id,msg_type,rpt_type,time_stamp,sp1,shelf,card,type,
      "FNN",fields);
  }
}

event DS3MUCBT1 {
  seqnum      "[0-9] + {}"
  jul         "[0-9] + {}"
  site_id     "... {}"
  msg_type    "M3A" {}
  rpt_type    ": D" {}
  time_stamp  MDY_HMS {}
  sp1         " " {}
  shelf       "... {}"
  comma       "," {}
  card         "[0-9]+ {}"
  type         " 15" {}
  par          " CBT" {}
  fields      "[0-9]+ {}"
  eom         "[\r\n]*"
  {
    ds3(seqnum,jul,site_id,msg_type,rpt_type,time_stamp,sp1,shelf,card,type,
  }
}

```

TABLE 4 (continued)

```
        "CNN",fields);
    }

event DS3MUCBTdashnoCOMPL {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "M3A {}"
    rpt_type    ": D {}"
    time_stamp  MDY_HMS {}
    sp1         " " {}
    shelf       "... {}"
    comma       "," {}
    card         "[0-9]+ {}"
    type         " 15" {}
    inf          " CBT" {}
    dash         " - " {}
    fields      "[0-9 %\r\n]+ {}"
    eom          "[\r\n]*"
{
    ds3 (seqnum,jul,site_id,msg_type,rpt_type,time_stamp,sp1,shelf,card,type,
          "CDN",fields);
}
}

event OCSU3C12 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "OSU: P {}"
    time_stamp  MDY_HMS {}
    sp1         " 3C12 " {}
    shelf       "... {}"
    comma       "," {}
    port         "[0-9]" {}
    sp           " " {}
    fields      "[0-9,]+ {}"
    eom          "COMPL[\r\n]*"
{
    store ("ocsu", catPipe("130",seqnum,jul,site_id,shelf,port,"181",
                           time_stamp,gen3_compl(fields)));
}
}
```

TABLE 4 (continued)

```

event OCSU3CI2PN {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "OSU: PN {}"
    time_stamp  MDY_HMS {}
    sp1         "3C12" {}
    shelf       "... {}"
    fields      "[0-9,,-]+{}"
    eom         "COMPL[\r\n]*"
    {
        store ("ocsu", catPipe("131",seqnum,jul,site_id,shelf,"0","181",
                               time_stamp,fields));
    }
}

event OCSU3CI2PI {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "OSU: PI {}"
    time_stamp  MDY_HMS {}
    sp1         "3C12" {}
    shelf       "... {}"
    fields      "[0-9,,-]+{}"
    eom         "COMPL[\r\n]*"
    {
        store ("oesu", catPipe("132",seqnum,jul,site_id,shelf,"0","181",
                               time_stamp,fields));
    }
}

event OCSUOC3P {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "OSU: P {}"
    time_stamp  MDY_HMS {}
    sp1         "OC3" {}
    shelf       "... {}"
    comma       "," {}
    port        "[0-9]+ {}"
    sp          " " {}
    fields      "[0-9,,-]+{}"
    eom         "COMPL[\r\n]*"
    {
        store ("ocsu", catPipe("140",seqnum,jul,site_id,shelf,port,"181",
                               time_stamp,fields));
    }
}

```

TABLE 4 (continued)

```

        time_stamp,gen3_compl(fields)));
    }
}

event OCSUOC3PN {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "OSU: PN" {}
    time_stamp  MDY_HMS {}
    spl         "OC3" {}
    shelf       "... {}"
    fields      "[0-9,]-]+{}"
    eom         "COMPL[\r\n]*"
    {
        store ("ocsu", catPipe("141",seqnum,jul,site_id,shelf,"0","181",
                               time_stamp,fields));
    }
}

event OCSUOC3PI {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "OSU: PI" {}
    time_stamp  MDY_HMS {}
    spl         "OC3" {}
    shelf       "... {}"
    fields      "[0-9,]-]+{}"
    eom         "COMPL[\r\n]*"
    {
        store ("ocsu", catPipe("142",seqnum,jul,site_id,shelf,"0","181",
                               time_stamp,fields));
    }
}

event PERF_POLL {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "CN" {}
    eq_type     "[0-9]" {}
    rpt_type    ": E[EW]" {}
    time_stamp  MDY_HMS {}
    shelf       "[0-9][0-9]" {}
    comma       "," {}
    card        "[0-9][0-9]" {}
}

```

TABLE 4 (continued)

```

dir      "[EW][0-9]+" {}
fields   "[0-9? ]+" {}
eom      "COMPL[\r\n]*"
{
  store ("esfmu", catPipe("10",seqnum,jul,site_id,conv_shelf(eq_type,
    shelf),card,"87",time_stamp,
    cat(eq_type,shelf),dir,pp_fill(fields,"6")));
}
}

event PERF_CSU {
  seqnum   "[0-9] + " {}
  jul      "[0-9] + " {}
  site_id  "... " {}
  msg_type "CN" {}
  eq_type   "[0-9]" {}
  rpt_type  ": EC" {}
  time_stamp MDY_HMS {}
  shelf    "[0-9][0-9]" {}
  comma    "," {}
  card     "[0-9][0-9]" {}
  status   "[0-9?]+ " {}
  fields   "[0-9?]+ " {}
  eom      "COMPL[\r\n]*"
{
  store ("esfmu",catPipe("20",seqnum,jul,site_id,conv_shelf(eq_type,shelf),card,"87",
    time_stamp,cat(eq_type,shelf),"E",csufill_field-list(fields,"4")));
}
}

event GEN315DASHCOMPL0 {
  seqnum   "[0-9] + " {}
  jul      "[0-9] + " {}
  site_id  "... " {}
  msg_type "CN" {}
  eq_type   "[0-9]" {}
  rpt_type  ": 15" {}
  time_stamp MDY_HMS {}
  shelf    "[0-9][0-9]" {}
  comma    "," {}
  card     "[0-9][0-9]" {}
  dir      "[EW]" {}
  type     "[0]" {}
  dash    " - " {}
  strt    "0-9]+ " {}
  sp      " " {}
  fields   "[0-9,---]+ " {}
}

```

TABLE 4 (continued)

```

com      "COMPL[\r\n]*"
{
}
}

event GEN315INCOMMNDASHNOCOMPL0 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "CN" {}
    eq_type     "[0-9]" {}
    rpt_type    ": 24" {}
    time_stamp  MDY_HMS {}
    shelf       "[0-9][0-9]" {}
    comma       "," {}
    card        "[0-9][0-9]" {}
    dir         "[EW]" {}
    type        "[0]" {}
    sp          " " {}
    fields      "[0-9],-]+{}"
    eom         "[\r\n]*"
{
}
}

event GEN324hourcompl {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "CN" {}
    eq_type     "[0-9]" {}
    rpt_type    ": 24" {}
    time_stamp  MDY_HMS {}
    shelf       "[0-9][0-9]" {}
    comma       "," {}
    card        "[0-9][0-9]" {}
    dir         "[EW]" {}
    type        "[01]" {}
    fields      "[0-9?,-]+{}"
    eom         "COMPL[\r\n]*"
{
}
}

event GEN324dash {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
}

```

TABLE 4 (continued)

```

site_id      "..." {}
msg_type     "CN" {}
eq_type      "[0-9]" {}
rpt_type     ": 24" {}
time_stamp   MDY_HMS {}
shelf        "[0-9][0-9]" {}
comma        "," {}
card         "[0-9][0-9]" {}
dir          "[EW]" {}
type         "[01]" {}
dash         "-" {}
strt         "[0-9]+ " {}
sp           " " {}
fields       "[0-9,--]+"
com          "COMPL[\r\n]*"
{
}
}

event GEN324incomplete {
    seqnum      "[0-9] + "
    jul         "[0-9] + "
    site_id    "..." {}
    msg_type    "CN" {}
    eq_type     "[0-9]" {}
    rpt_type    ": 24" {}
    time_stamp  MDY_HMS {}
    shelf       "[0-9][0-9]" {}
    comma      ","
    card        "[0-9][0-9]" {}
    dir         "[EW]" {}
    type        "[01]" {}
    sp          " " {}
    fields      "[0-9,---]+"
    com         "[\r\n]*"
{
}
}

event GEN324INCOMDASHNOCOMPL1 {
    seqnum      "[0-9] + "
    jul         "[0-9] + "
    site_id    "..." {}
    msg_type    "CN" {}
    eq_type     "0-9" {}
    rpt_type    ": 24" {}
    time_stamp  MDY_HMS {}
}

```

TABLE 4 (continued)

```

shelf      "[0-9][0-9]" {}
comma     ","
card      "[0-9][0-9]" {}
dir       "[EW]" {}
type      "[01]" {}
dash      "- " {}
strt      "[0-9]+"
fields   "[0-9,---]+"
com      "[\r\n]*"
{
}
}

event GEN315COMPL0 {
  seqnum    "[0-9] +"
  jul       "[0-9] +"
  site_id   "..."
  msg_type  "CN"
  eq_type   "[0-9]"
  rpt_type  ": 15"
  time_stamp MDY_HMS {}
  shelf      "[0-9][0-9]" {}
  comma     ","
  card      "[0-9][0-9]" {}
  dir       "[WE]" {}
  typ       "[0]" {}
  fields   "[0-9 ?---]+"
  eom      "COMPL[\r\n]*"
{
}
}

event DS3MUPARCOMPL24 {
  seqnum    "[0-9] +"
  jul       "[0-9] +"
  site_id   "..."
  msg_type  "M3A"
  rpt_type  ": 24"
  time_stamp MDY_HMS {}
  spl       " "
  shelf     "... "
  comma     ","
  card      "[0-9] +"
  type      "24"
  inf       "PAR"
  dash      "- "
  fields   "[0-9 %.-]+"
}

```

TABLE 4 (continued)

```

eom1      "COMPL" {}
eom      "[\r\n]*"
{
}
}

event DS3MUPARdashnoCOMPL24 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "M3A" {}
    rpt_type    ": 24" {}
    time_stamp  MDY_HMS {}
    spl         " " {}
    shelf       "... {}"
    comma       "," {}
    card        "[0-9] +" {}
    type        "24" {}
    dash        "- " {}
    inf         "PAR" {}
    fields      "[0-9 %.-]+ {}"
    eom        "[\n\r]*"
{
}
}

event DS3MUFEPCOMPL24 {
    seqnum      "[0-9] + {}"
    jul         "[0-9] + {}"
    site_id     "... {}"
    msg_type    "M3A" {}
    rpt_type    ": 24" {}
    time_stamp  MDY_HMS {}
    spl         " " {}
    shelf       "... {}"
    comma       "," {}
    card        "[0-9] +" {}
    type        "24" {}
    inf         "FEP" {}
    dash        "- " {}
    fields      ".-*COMPL[\r\n]*"
{
}
}

event DS3MUFEPI24 {
    seqnum      "[0-9] + {}"

```

TABLE 4 (continued)

```

    jul      "[0-9]+ {}"
    site_id  "... {}"
    msg_type "M3A" {}
    rpt_type "; 24" {}
    time_stamp MDY_HMS {}
    spl      " " {}
    shelf    "... {}"
    comma   ";" {}
    card    "[0-9]+ {}"
    type    "24" {}
    par     "FEP" {}
    fields  "[0-9]+ {}"
    eom    "[\r\n]*"
    {
    }
}

event DS3MUCBT124 {
    seqnum  "[0-9]+ {}"
    jul      "[0-9]+ {}"
    site_id  "... {}"
    msg_type "M3A" {}
    rpt_type "; 24" {}
    time_stamp MDY_HMS {}
    spl      " " {}
    shelf    "... {}"
    comma   ";" {}
    card    "[0-9]+ {}"
    type    "24" {}
    par     "CBT" {}
    fields  "[0-9]+ {}"
    eom    "[\r\n]*"
    {
    }
}

event DS3MUCBTdashnoCOMPL24 {
    seqnum  "[0-9]+ {}"
    jul      "[0-9]+ {}"
    site_id  "... {}"
    msg_type "M3A" {}
    rpt_type "; 24" {}
    time_stamp MDY_HMS {}
    spl      " " {}
    shelf    "... {}"
    comma   ";" {}
    card    "[0-9]+ {}"
}

```

TABLE 4 (continued)

```
type  "24" {}
inf   "CBT" {}
dash  "- " {}
fields "[0-9].*" {}
eom   "[\r\n]*"
{
}
}
```

TABLE 4 (continued)

ESFMU

Not all ESFMUs are included in the ODB. Only those ESFMUs which appear in the SEGMENT and TRUNK topology tables are included. A separate table containing these ESFMUs is created and is used to filter out unwanted ESFMUs. This table is called the `monitor_ctl` table.

1. `monitor_ctl`

This table is used to control which ESFMUs are included in the ODB.

| | | |
|----------------------------|-------------------------|---|
| <code>site_id</code> | <code>char(3)</code> | |
| <code>equip_alpha</code> | <code>char(3)</code> | The three character designator for the equipment. For esfmu, this represents the shelf, the first character is blank. |
| <code>port_nbr</code> | <code>numeric(6)</code> | Port number. For esfmu, this is the card. |
| <code>equip_code</code> | <code>smallint</code> | Equipment code. 87=esfmu |
| <code>src_cntl_code</code> | <code>smalling</code> | Source Control Code. 0=production, 1=DDN Certification |
| <code>circ_id</code> | <code>char(12)</code> | Circuit ID |
| <code>eff_dt</code> | <code>datetime</code> | Effective date and time. |
| <code>disc_dt</code> | <code>datetime</code> | Discontinue date and time. The date and time that this monitor point is no longer effective. |

2. `odb_sites`

This table is a list of all sites and relates the `site_id` with the `site_nbr`.

| | | |
|-----------------------|--------------------------|------------------|
| <code>site_id</code> | <code>char(3)</code> | Site identifier. |
| <code>site_nbr</code> | <code>numeric(12)</code> | Site number. |

3. `odb_equip_codes`

This table is used to relate the equipment code with the equipment alpha identifier and the `eq_typ`.

| | | |
|--------------------------|-----------------------|---|
| <code>equip_code</code> | <code>smalling</code> | Equipment code. 87=esfmu |
| <code>equip_alpha</code> | <code>char(3)</code> | The three character designator for the equipment. For esfmu, this represents the shelf, the first character is blank. |
| <code>eq_typ</code> | <code>char(3)</code> | Equipment type. For esfmu, this would be "CNx" where x is the esfmu number from 0 through n. |

4. esfmu_reports

For each ESFMU performance message is rd and is in the monitor_cntl table, there is an entry in the esfmu_reports table. Performance nominal messages are expanded out. The actual performance indicators, such as line errored seconds etc., are stored in the esfmu_detail table. This allows for data compression since repeated performance indicators are only stored once.

| | | |
|-------------|-------------|--|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. For esfmu, this represent the shelf, the first character is blank. |
| port_nbr | numeric(6) | Port number. For esfmu, this is the card. |
| slot_side | char(1) | Slot side. E=East, W=West. |
| rpt_dt | datetime | Report date and time. The date and time as "stamped" by the site controller. |
| dtl_key | numeric(15) | The key to the esfmu_detail table. |
| ld_dt | datetime | The date and time that the rwo was inserted into the table. |

TABLE 5 (continued)

5. esfmu_detail

| | | |
|--------------|-------------|--|
| dtl_key | numeric(15) | |
| gen | smalling | Generation of the ESFMU (2=GEN 2, 3=GEN 3, duh!) |
| ssi | int | seconds in interval |
| mms | int | Maintenance Mode Seconds |
| niss | int | Not-in_service seconds |
| res | int | Raw errored seconds. |
| les | int | Line Errored seconds |
| rses | int | Raw Severely errored seconds. |
| lses | int | Line Severely Errored Seconds |
| oof | int | Out of frame. |
| fss | int | Failed signal state. |
| csu_ss | int | CSU status byte |
| csu_esf_err | int | CSU ESF errors |
| scu_synch_es | int | CSU synchronous errored seconds |
| csu_synch_fs | int | CSU synchronous failed seconds |
| llos | int | Line Loss of signal seconds |
| lcv | int | Line Code Violation |
| psas | int | Path SEF/AIS seconds |
| paiss | int | Path Alarm Indication Signal Seconds |
| pcss | int | Path Controlled Slip Seconds |
| pes | int | Path Errored Seconds |
| pses | int | Path Severely Error Seconds |
| puas | int | Path Unavailable Seconds |
| pcv | int | Path Code Violations |
| fe_pcv | int | Path Code Violations - far end |
| fe_pes | int | Path Errored Seconds - far end |
| pesa | int | Path Errored Seconds - type A |
| fe_pesa | int | Path Errored Seconds - type A, far end |
| pesb | int | Path Errored Seconds - type B |
| fe_pesb | int | Path Errored Seconds - type B, far end |
| fe_pses | int | Path Severely Errored Seconds - far end |
| fe_psefs | int | Path Severely Errored Frame Seconds - far end |
| fe_pcsl | int | Path Controlled Slip Seconds - far end |
| fe_paus | int | Path Unavailable Seconds - far end |
| fe_les | int | Line Errored Seconds - far end |
| lfec | int | Line Failure Count |
| pfec | int | Path Failure Count |
| fe_pfec | int | Path Failure Count - far end |

TABLE 5 (continued)

6. esfmu_exceptions

This table contains esfmu messages which are in error.

| | | |
|-------------|--------------|---|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. For esfmu, this represents the shelf, the first character is blank. |
| port_nbr | numeric(6) | Port number. For esfmu, this is the card. |
| slot_side | char(1) | Slot side. E=East, W=West. |
| rpt_dt | datetime | Report date and time. The date and time as "stamped" by the site controller. |
| ld_dt | datetime | The date and time that the row was inserted into the table. |
| msg_txt | varchar(255) | Actual text of the message as it was received from the LSE. |

7. esfmu_cert_summary

This includes the summary of both gen2 and gen3 data for those monitor control points which have a source control code of "1".

| | | |
|-------------|------------|---|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. For esfmu, this represents the shelf, the first character is blank. |
| port_nbr | numeric(6) | Port number. For esfmu, this is the card. |
| slot_side | char(1) | Slot side. E=East, W=West. |
| rpt_dt | datetime | Report date and time. The date and time as "stamped" by the site controller. The time is set to "00:00:00". |
| ld_dt | datetime | The date and time that the row was inserted into the table. |
| ttl_rpts | int | The number of reports included in the daily summary. |
| rpt_stts | smallint | 0=no reports for the day, 1=complete, 2=incomplete (not all 15-min intervals reporting) |
| les | int | Line errored seconds. |
| lses | int | Line severe errored seconds. |
| res | int | Raw errored seconds. |
| oof | int | Out of frame. |
| fss | int | Failed signal state. |
| csu_es_err | int | Customer Service Unit, errored seconds. |
| csu_ses | int | Customer Service Unit, severely errored seconds. |
| csu_fss | int | Customer Service Unit, failed signal state. |

TABLE 5 (continued)

BBDXC**1. bbdxc_reports**

Each performance message from the Alcatel 1633 Broadband DXC is stored in the ODB. Unlike the ESFMUs, no attempt is made to filter out messages. A range of port numbers is stored in a single row in the bbdxc_reports table. The first port number in a range is "port_nbr". The last port number in a range is "end_port_nbr". The port_nbr and the end_port_nbr may be the same.

| | | |
|--------------|-------------|--|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. |
| port_nbr | numeric(6) | Port number. |
| rpt_dt | datetime | Report date and time. The date and time as "stamped" by the site controller. |
| dtl_key | numeric(15) | The key to the bbdxc_detail table. A zero indicates performance nominal and a one (1) indicates a performance incomplete (all NULL). |
| ld_dt | datetime | The date and time that the row was inserted into the table. |
| end_port_nbr | numeric(6) | The last port number in a range of port numbers. |

TABLE 5 (continued)

2. bbdxc_detail

The detail key is used internally by ODB to accomplish data compression and joins between the bbdxc_detail and the bbdxc_reports.

| | | |
|---------|-------------|--|
| dtl_key | numeric(15) | Unique identifier for the <u>bbdxc_detail</u> table. |
| lcv | int | Line code violation |
| pcv | int | Path code violation |
| pfpe | int | Path frame bit errors |
| poofe | int | Path out of frame errors |
| les | int | Line errored seconds |
| pes | int | Path errored seconds |
| lses | int | Line severely errored seconds |
| pses | int | Path severely errored seconds |
| pcses | int | Path consecutive severely errored seconds |
| puas | int | Path unavailable seconds |
| ppsc | int | Path protection switch counts |
| ppsd | int | Path protection switch duration |
| ppes | int | Path parity errored seconds |
| pjes | int | Path frame errored seconds |
| paiss | int | path alarm indication signal seconds |
| pfbe | int | Path frame bit errors |
| pepe | int | Path C-bit parity errors |
| llos | int | Line loss of signal seconds |

3. bbdxc_exceptions

This table contains bbdxc messages which are in error.

| | | |
|--------------|--------------|--|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. |
| port_nbr | numeric(6) | Port number. |
| rpt_dt | datetime | Report date and time. The date and time as "stamped" by the site controller. |
| ld_dt | datetime | The date and time that the row was inserted into the table. |
| end_port_nbr | numeric(6) | The last port number in a range of port numbers. |
| msg_txt | varchar(255) | Actual text of the message as it was received from the LSE. |

TABLE 5 (continued)

DS3MU**1. ds3mu_reports**

| | | |
|-------------|-------------|--|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. |
| port_nbr | numeric(6) | Port number. |
| rpt_dt | datetime | Report date and time. The date and time as 'stamped' by the site controller. |
| dtl_key | numeric(15) | The key to the ds3mu_detail table. A zero indicates performance nominal and a one (1) indicates a performance incomplete (all NULL). |
| ld_dt | datetime | The date and time that the rwo was inserted into the table. |

2. ds3mu_detail

Near-end M23, near-end C-bit and far-end C-bit application fields are included in the same table.

TABLE 5 (continued)

| | | | |
|----------|---------|--|--|
| dtl_key | int | | |
| ssi | int | Seconds in Interval | |
| nis | char(2) | Not in service seconds | 0-900 for 15 min/0-86400 for 24 hr |
| mm | char(3) | Monitoring Mode | 0-900 for 15 min/0-86400 for 24 hr |
| cb | char(2) | C-bit indicator | PB=Parity bit/FB-Frame Bit |
| cf | char(2) | Change of format | CBT for c-bit/M23 for m23 |
| ipi | int | Incomplete Period Indicator | FC=form. change/NC=no change |
| les | int | Line Errored Seconds | P1 for incom/OK for complete |
| llos | int | Line Loss of Signal Seconds | 0-900 for 15 min/0-86400 for 24 hr |
| pcv | int | Path Code Violation | 0-900 for 15 min/0-86400 for 24 hr |
| pes | int | Path Errored Seconds | 0-65535 for 15 min/0-4194303 for 24 hr |
| pesa | int | Path Errored Seconds Type A | 0-900 for 15 min/0-86400 for 24 hr |
| pesb | int | Path Errored Seconds Type B | 0-900 for 15 min/0-86400 for 24 hr |
| pses | int | Path Severely Errored Seconds | 0-900 for 15 min/0-86400 for 24 hr |
| pefs | float | Path Percent Err. Free Seconds | 0-900 for 15 min/0-86400 for 24 hr |
| pbber | float | Path Background Bit Error Rate | 000.00% to 100.0% |
| | | | 00-99 for mm/-06 to -11 ee 15 min/-06 |
| | | | to -13 ee 24 hr |
| puas | int | Path Unavailable Seconds | 0-900 for 15 min/0-86400 for 24 hr |
| pavl | float | Path Percent Err. Availability | 0.00% to 100.0% |
| psefs | int | Path Severely Err. Frame Seconds | 0-900 for 15 min/0-86400 for 24 hr |
| paiss | int | Path Alarm Indicat Signal Secs | 0-900 for 15 min/0-86400 for 24 hr |
| ne_pcv | int | Path Code Violation - near end | 0-65535 for 15 min/0-4194303 for 24 hr |
| ne_pes | int | Path Errored Seconds - near end | 0-900 for 15 min/0-86400 for 24 hr |
| ne_pesa | int | Path Errored Seconds Type A - near end | 0-900 for 15 min/0-86400 for 24 hr |
| ne_pesb | int | Path Errored Seconds Type B - near end | 0-900 for 15 min/0-86400 for 24 hr |
| ne_pses | int | Path Severely Errored Seconds - near end | 0-900 for 15 min/0-86400 for 24 hr |
| ne_pefs | int | Path Percent Err. Free Second - near end | 0-900 for 15 min/0-86400 for 24 hr |
| ne_ppber | int | Path Background Bit Error Rate - near end | 000.00% to 100.0% |
| | | | 00-99 for mm/-06 to -11 ee 15min/-06 |
| | | | to -13 ee 24 hr |
| ne_puas | int | Path Unavailable Seconds - near end | 0-900 for 15 min/0-86400 for 24 hr |
| ne_pavl | float | Path Percent Err. Availability - near end | 0.00.00% to 100.0% |
| fe_pcv | int | Path Code Violation - far end | 0-65535 for 15 min/0-4194303 for 24 hr |
| fe_pes | int | Path Errored Seconds - far end | 0-900 for 15 min/0-86400 for 24 hr |
| fe_pesa | int | Path Errored Seconds Type A - far end | 0-900 for 15 min/0-86400 for 24 hr |
| fe_pesb | int | Path Errored Seconds Type B - far end | 0-900 for 15 min/0-86400 for 24 hr |
| fe_pses | int | Path Severely Errored Seconds - far end | 0-900 for 15 min/0-86400 for 24 hr |
| fe_pefs | float | Path Percent Err. Free Seconds - far end | 0-900 for 15 min/0-86400 for 24 hr |
| fe_ppber | float | Path Background Bit Error Rate - far end | 000.00% to 100.0% |
| | | | 00-99 for mm/-06 to -11 ee 15 min/-06 |
| | | | to -13 ee 24 hr |
| fe_paus | int | Path Unavailable Seconds - far end | 0-900 for 15 min/0-86400 for 24 hr |
| fe_pavl | float | Path Percent Err. Availability | 0.0% to 100.0% |
| fe_psefs | int | Path Severely Err. Frame Seconds - far end | 0-900 for 15 min/0-86400 for 24 hr |
| fe_paiss | int | Path Alarm Indicat Signal Secs - far end | 0-900 for 15 min/0-86400 for 24 hr |
| | | | 0-900 for 15 min/0-86400 for 24 hr |

TABLE 5 (continued)

3. **ds3mu_exceptions**

This table contains ds3mu messages which are in error.

| | | |
|-------------|--------------|--|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. |
| port_nbr | numeric(6) | Port number. |
| rpt_dt | datetime | Report date and time. The date and time as a "stamped" by the site controller. |
| ld_dt | datetime | The date and time that the row was inserted into the table. |
| msg_txt | varchar(255) | Actual text of the message as it was received from the LSE. |

E1MU

The alpha designator is a three character shelf name. The port number is a function of the card number (1-15) and the monitoring point number (1-16)

$$\text{Port\#} = ((\text{card number} - 1) * 16) + \text{monitoring point number}$$

Therefore port numbers 1-15 are the 15 monitoring points on the first card. Port number 17 would be card 2, monitoring point number 1 and so on.

1. **e1mu_reports**

| | | |
|--------------|-------------|---|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. |
| port_nbr | numeric(6) | Port number. |
| rpt_dt | datetime | Report date and time. The date and time as a "stamped" by the site controller. |
| dtl_key | numeric(15) | The key to the elmu_detail table. A zero indicates performance nominal and a one (1) indicates a performance incomplete (all NULL). |
| ld_dt | datetime | The date and time that the row was inserted into the table. |
| end_port_nbr | numeric(6) | The last port number in a range of port numbers. |

TABLE 5 (continued)

2. **elmu_detail**

| dtl_key | | |
|---------|-------|-----------------------------------|
| aiss | int | Alarm Indication Signal Seconds |
| pbber | float | Path Background Block Error Ratio |
| pber | float | Path Bit Error Ratio |
| lbes | int | Line Burst Errored Seconds |
| pbes | int | Path Burst Errored Seconds |
| rbes | int | Remote Burst Errored Seconds |
| code | int | Code Error |
| crc | int | Cyclic Redundancy Checksum error |
| les | int | Line Errored Seconds |
| pes | int | Path Errored Seconds |
| r_es | int | Remote Errored Seconds |
| ess | int | Service Errored Seconds |
| fas | int | Frame Alignment Signal error |
| lofs | int | Loss Of Frame Seconds |
| llos | int | Line Loss Of Signal Seconds |
| prbs | int | Pseudo Random Bit Sequence error |
| rais | int | Remote Alarm Indication Seconds |
| rebe | int | Remote End Block Error |
| lses | int | Line Severely Errored Seconds |
| pses | int | Path Severely Errored Seconds |
| r_ses | int | Remote Severely Errored Seconds |
| sses | int | Service Severely Errored Seconds |
| luas | int | Line Unavailable Seconds |
| puas | int | Path Unavailable Seconds |
| ruas | int | Remote Unavailable Seconds |
| suas | int | Service Unavailable Seconds |

3. **elmu_exceptions**

This table contains elmu messages which are in error.

| | | |
|--------------|--------------|--|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment. |
| port_nbr | numeric(6) | Port number. |
| rpt_dt | datetime | Report date and time. The date and time as "stamped" by the site controller. |
| ld_dt | datetime | The date and time that the row was inserted into the table |
| end_port_nbr | numeric(6) | The last port number in a range of port numbers. |
| msg_txt | varchar(255) | Actual text of the message as it was received from the LSE. |

OCSU

TABLE 5 (continued)

1. ocsu_reports

| | | |
|--------------|-------------|---|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment (shelf name) |
| mux_lvl | char(4) | MUX level. "3C12"=Imbedded VC-12 data, "OC3"=OC3 data |
| port_nbr | numeric(6) | Port number. |
| | | VC-12: |
| | | 1-63 = Facility side VC12s 1-63 |
| | | 64-126 = Terminal side VC12s 1-63 |
| | | OC3: |
| | | 1 = Facility side OC3 |
| | | 2 = Terminal side OC3 |
| rpt_dt | datetime | Report date and time. The date and time as "stamped" by the site controller. |
| dtl_key | numeric(15) | The key to the elmu_detail table. A zero indicates performance nominal and a one (1) indicates a performance incomplete (all NULL). |
| ld_dt | datetime | The date and time that the row was inserted into the table. |
| end_port_nbr | numeric(6) | The last port number in a range of port numbers. (see port_nbr above for values.) |

2. oscu_detail

| | | |
|---------|-------------|---|
| dtl_key | numeric(15) | |
| es | int | VC-12 Errorred seconds, mux_lvl = "3C12" |
| ses | int | VC-12 Severely errored seconds, mux_lvl = "3C12" |
| uas | int | VC-12 Unavailable seconds, mux_lvl = "3C12" |
| eb | int | VC-12 Errorred blocks, mux_lvl = "3C12" |
| fe_es | int | VC-12 Errorred seconds, far end, mux_lvl = "3C12" |
| fe_ses | int | VC-12 Severely errored seconds, far end, mux_lvl = "3C12" |
| fe_uas | int | VC-12 Unavailable seconds, far end, mux_lvl = "3C12" |
| fe_eb | int | VC-12 Errorred blocks, far end, mux_lvl = "3C12" |
| s_es | int | OC3 Section errored seconds, mux_lvl = "OC3 " |
| s_ses | int | OC3 Section severely errored seconds, mux_lvl = "OC3 " |
| sefs | int | OC3 Severely errored framing seconds, mux_lvl = "OC3 " |
| s_cv | int | OC3 Section coding violations, mux_lvl = "OC3 " |
| pcv | int | OC3 Path coding violations, mux_lvl = "OC3 " |

TABLE 5 (continued)

3. ocsu_exceptions

| | | |
|---------------------|---------------------|--|
| site_id | char(3) | Site identifier. |
| equip_alpha | char(3) | The three character designator for the equipment (shelf name) |
| mux_lvl | char(4) | MUX level. "3C12"=Imbedded VC-12 data, "OC3"=OC3 data |
| port_nbr | numeric(6) | Port number.
VC-12:
1-63 = Facility side VC12s 1-63
64-126 = Terminal side VC12s 1-63
OC3:
1 = Facility side OC3
2 = Terminal side OC3 |
| rpt_dt | datetime | Report date and time. The date and time as "stamped" by the site controller. |
| dtl_key | numeric(15) | The key to the <u>elmu_detail</u> table. A zero indicates performance nominal and a one (1) indicates a performance incomplete (all NULL). |
| ld_dt | datetime | The date and time that the row was inserted into the table. |
| end_port_nbr | numeric(6) | The last port number in a range of port numbers. (see <u>port_nbr</u> above for values.) |
| msg_txt | varchar(255) | Actual text of the message as it was received from the LSE. |

TABLE 5 (continued)

Conclusion

While various embodiments of the present invention have been described above, it should be understood that they have been presented by the way of example only, and not limitation. If it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined in the appended claims. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What Is Claimed Is:

1. A system for storing and retrieving performance and topology data of a telecommunications network, comprising:
 - 5 an open database management system for storing performance and topology data;
 - a network performance data parser for receiving network performance messages and parsing said network performance messages into performance data;
 - 10 a network performance data loader for loading said performance data into said open database management system;
 - a topology data loader for loading topology data into said open database management system; and
 - 15 a sonet data loader for loading sonet data into said open database management system.
2. The system according to claim 1, further comprising:
 - 20 a plurality of parsing rules to define a mapping between said network performance messages and said performance data;
 - one or more performance data files in which said network performance data parser stores said performance data, and from which said network performance data loader retrieves said performance data to load into said open database management system;
 - 25 a topology data file in which a network real time topology system stores said topology data, and from which said topology data loader retrieves said topology data to load into said open database management system; and
 - a sonet data file in which a sonet device stores said sonet data, and from which said sonet data loader retrieves said sonet data to load into said open database management system means.

3. The system according to claim 2, further comprising one or more tracking file to store file information on said performance data files.
4. The system according to claim 2, further comprising a placeholder file to store a position in said performance data file at which said network performance data loader is currently processing said performance data file.
5. A method for storing and retrieving network performance data of a telecommunications network, comprising the steps of:
 - (1) receiving said network performance data; and
 - (2) loading said network performance data in an open database management system.
6. The method according to claim 5, wherein step (1) comprises :
 - (1.a) reading a network performance data message;
 - (1.b) determining whether said network performance data message matches a parsing rule;
 - (1.c) parsing said network performance data message according to said parsing rule, if it is determined in step (1.b) that said network performance data message matches said parsing rule;
 - (1.d) reformatting said network performance data message into a reformatted performance message if step (1.c) parses said network performance data message; and
 - (1.e) writing said reformatted performance message to a performance data file if step (1.d) reformates said network performance data message.
7. The method of claim 6, wherein step (1.e) comprises :
 - (1.e.1) identifying a performance data file with the most recent time/date stamp;

(1.e.2) determining whether said performance data file that was identified in step (1.e.1) exists;

(1.e.3) creating said performance data file if it is determined in step (1.e.2) that said performance data file does not exist;

5 (1.e.4) writing said reformatted performance message to said performance data file;

(1.e.5) determining whether said performance data file is full;

(1.e.6) closing said performance data file if it is determined in step (1.e.5) that said performance data file is full; and

10 (1.e.7) creating a new performance data file if it is determined in step (1.e.5) that said performance data file is full.

8. The method of claim 7, wherein step (1.e) further comprises :

(1.e.8) identifying a tracking file with the most recent time/date stamp;

15 (1.e.9) determining whether said tracking file that was identified in step (1.e.8) exists;

(1.e.10) creating said tracking file if it is determined in step (1.e.8) that said tracking file does not exist;

(1.e.11) creating an entry in said tracking file; and

20 (1.e.12) updating said entry in said tracking file to reflect the writing of said reformatted performance message in step (1.e.4).

9. The method of claim 5, wherein step (2) comprises :

(2.a) connecting to said open database management system;

25 (2.b) retrieving a list of specific monitoring points in said telecommunications network;

(2.c) retrieving a data record from a performance data file, wherein said data record contains network performance data;

(2.d) determining a performance data type of said data record; and

30 (2.e) writing said data record to said open database management system.

10. The method of claim 9, wherein step (2.c) comprises :

- (2.c.1) opening a placeholder file;
- (2.c.2) reading an entry in said placeholder file that indicates said performance data file from which to read said data record;
- 5 (2.c.3) reading said data record from said performance data file;
- (2.c.4) determine whether said performance data file returned an end-of-file while performing step (2.c.3);
- (2.c.5) returning said data record if it is determined in step (2.c.4) that said performance data file did not return an end-of-file;
- 10 (2.c.6) closing and deleting said performance data file if it is determined in step (2.c.4) that said performance data file did return an end-of-file;
- (2.c.7) determining if a second performance data file exists if it is determined in step (2.c.4) that said performance data file did return an end-of-file;
- 15 (2.c.8) returning a first data record from said second performance data file if it is determined in step (2.c.4) that said performance data file did return an end-of-file;
- (2.c.9) updating said entry in said placeholder file to indicate said return of said data record from said performance data file or to indicate said return of said first data record from said second performance data file.

20

11. The method of claim 10, wherein step (2.c) further comprises :

- (2.c.10) opening a tracking file;
- 25 (2.c.11) retrieving a name of said second performance data file from said tracking file if it is determined in step (2.c.4) that said performance data file returned an end-of-file;
- (2.c.12) determining if said tracking file returns an end-of-file when retrieving said name of said second performance data file in step (2.c.11); and

30

(2.c.13) closing and deleting said tracking file if it is determined in step (2.c.12) that said tracking file returned an end-of-file.

12. The method for storing and retrieving topology and sonet data of a telecommunications network, comprising the steps of :

5 (1) receiving said topology data;
(2) loading said topology data in an open database management system;
(3) receiving said sonet data; and
(4) loading said sonet data in said open database management system.

10

13. The method of claim 12, wherein step (2) comprises :

15 (2.a) connecting to an open database management system;
(2.b) retrieving a data record from a topology data file, wherein said data record contains network topology data;
(2.c) determining whether said topology data file returns an end-of-file;
(2.d) determining a topology data type of said data record if it is determined in step (2.c) that said topology data file does not return an end-of-file; and
20 (2.e) writing said data record to said open database management system if it is determined in step (2.c) that said topology data file does not return an end-of-file.

25 14. The method of claim 12, wherein step (4) comprises :

(4.a) connecting to an open database management system;
(4.b) retrieving a data record from a sonet data file, wherein said data record contains network sonet data;
(4.c) determining whether said sonet data file returns an end-of-file;

(4.d) determining a sonet data type of said data record if it is determined in step (4.c) that said sonet data file does not return an end-of-file; and

5 (4.e) writing said data record to said open database management system if it is determined in step (4.c) that said sonet data file does not return an end-of-file.

15. A computer program product for use with a database management system, comprising :

10 a computer usable medium having computer readable program code means embodied in said medium for enabling a processor to store and retrieve performance and topology data of a telecommunications network, said computer program product having :

15 first computer readable program code means for enabling a processor to receive network performance messages;

second computer readable program code means for enabling a processor to receive topology data;

third computer readable program code means for enabling a processor to receive sonet data; and

20 fourth computer readable program code means for enabling a processor to store said network performance messages, said topology data, and said sonet data.

16. The computer program product according to claim 15, wherein said first computer readable program code means comprises :

25 network performance data parsing means for enabling a processor to receive said network performance messages and parse said network performance messages into performance data; and

network performance data loader means for enabling a processor to load said performance data into said fourth computer readable program code means.

17. The computer program product according to claim 16, wherein said network performance data parsing means further comprises :

5 a plurality of parsing rules to define a mapping between said network performance messages and said performance data;

10 one or more performance data files in which said network performance data parsing means enables a processor to store said performance data, and from which said network performance data loader means enables a processor to retrieve said performance data and load said performance data into said fourth computer readable program code means; and

15 one or more tracking files to store file information on said performance data files.

18. The computer program product according to claim 16, wherein said network performance data loader means comprises a placeholder file to store a position in said performance data file at which said network performance data loader means enables a processor to process said performance data file.

20 19. The computer program product according to claim 15, wherein said second computer readable program code means further comprises a topology data file in which a network real time topology system stores said topology data, and from which said second computer readable program code means enables a processor to retrieve said topology data and load said topology data into said fourth computer readable program code means.

25 20. The computer program product according to claim 15, wherein said third computer readable program code means further comprises a sonet data file in which a sonet device stores said sonet data, and from which said third computer readable program code means enables a processor to

retrieve said sonet data and load said sonet data into said fourth computer
readable program code means.

1/14

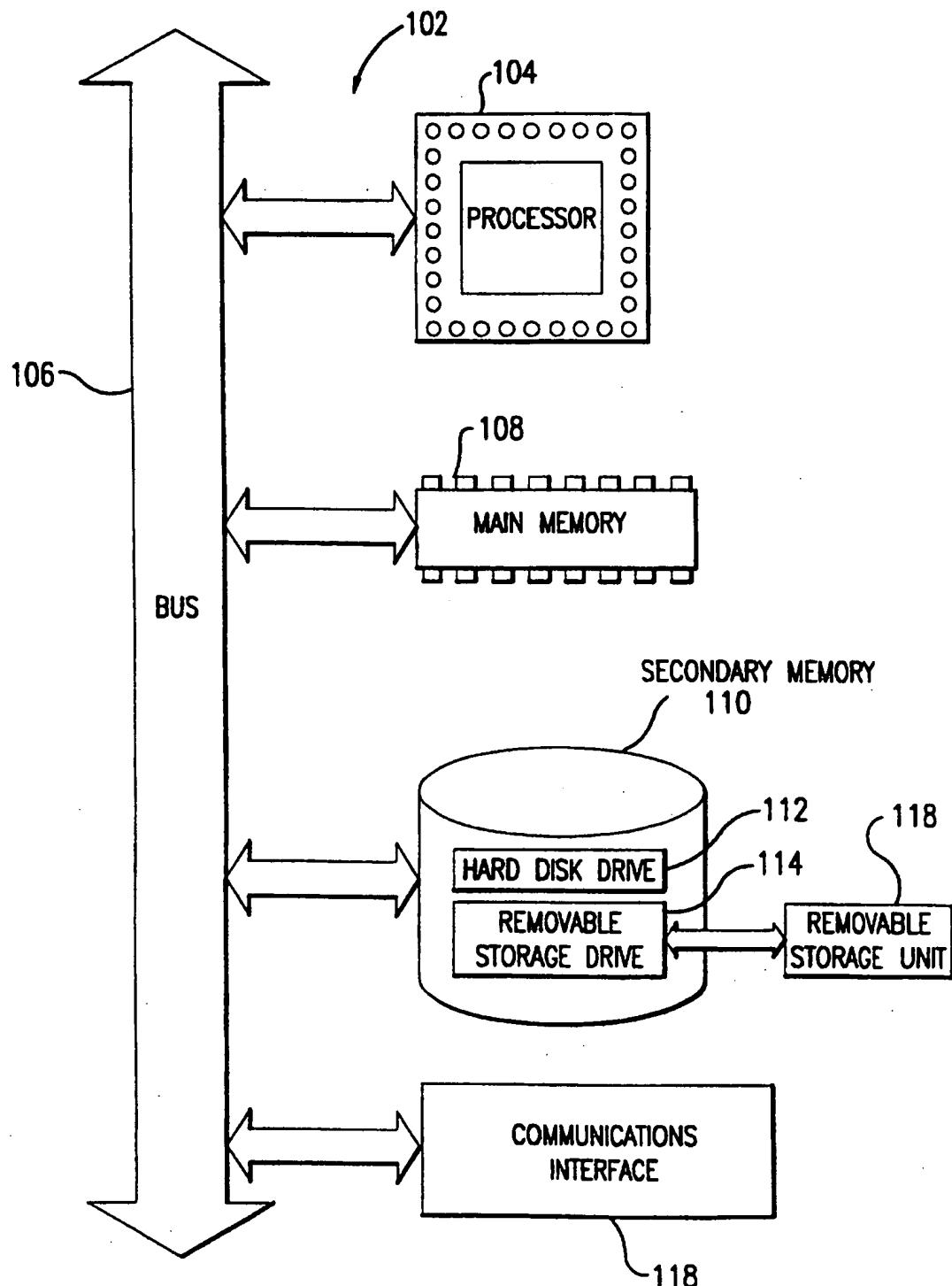


FIG.1

SUBSTITUTE SHEET (RULE 26)

2/14

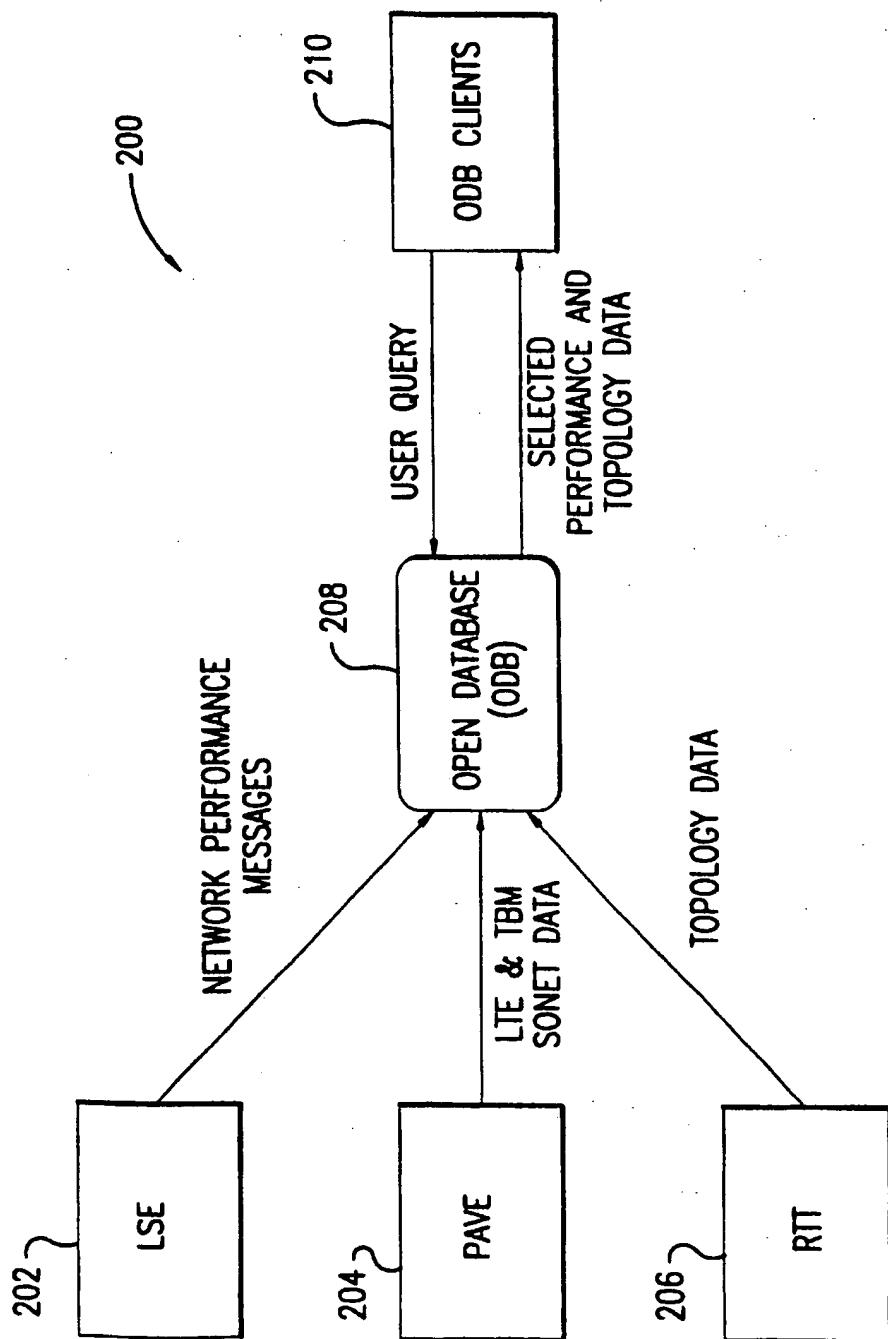


FIG.2

3/14

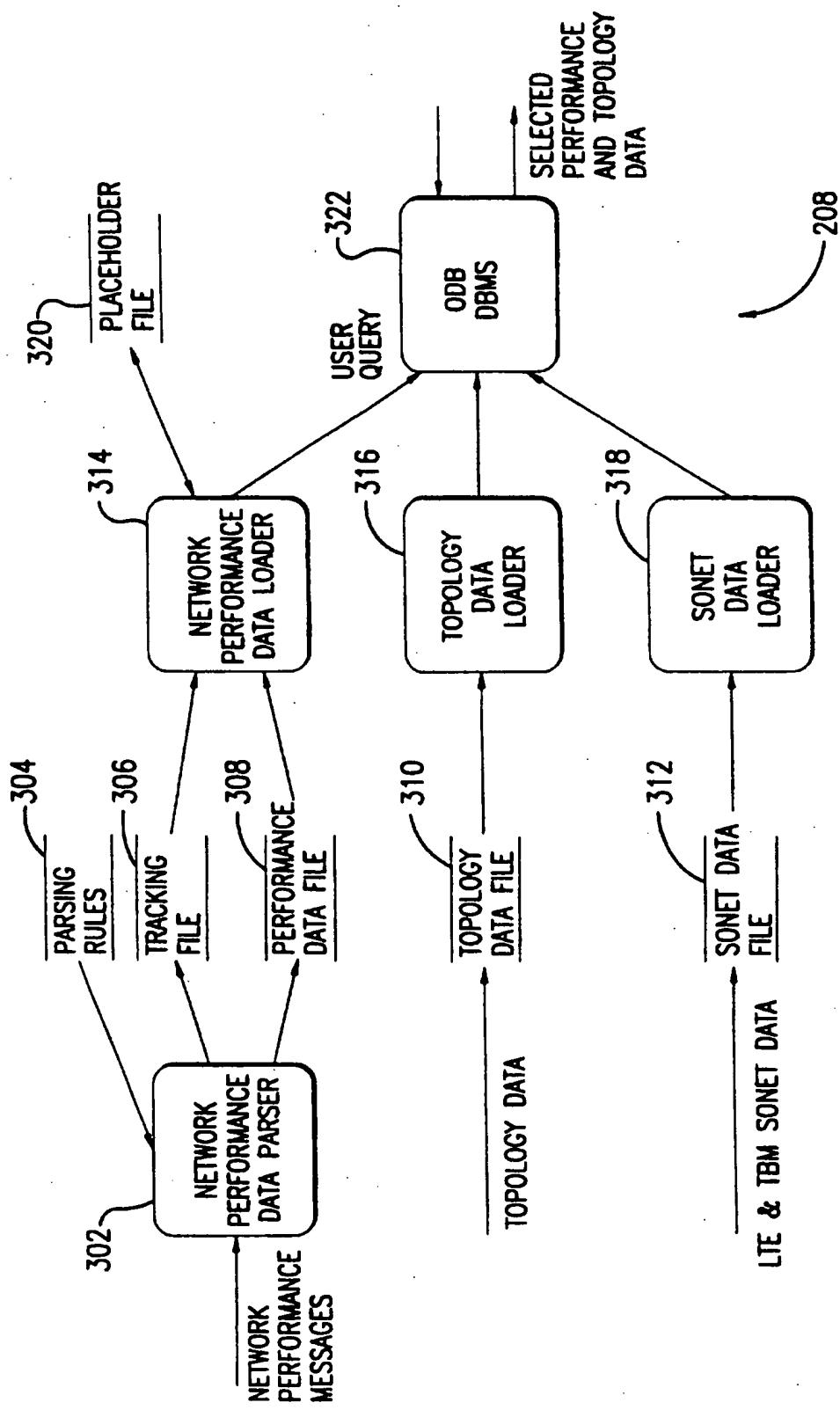


FIG.3

4/14

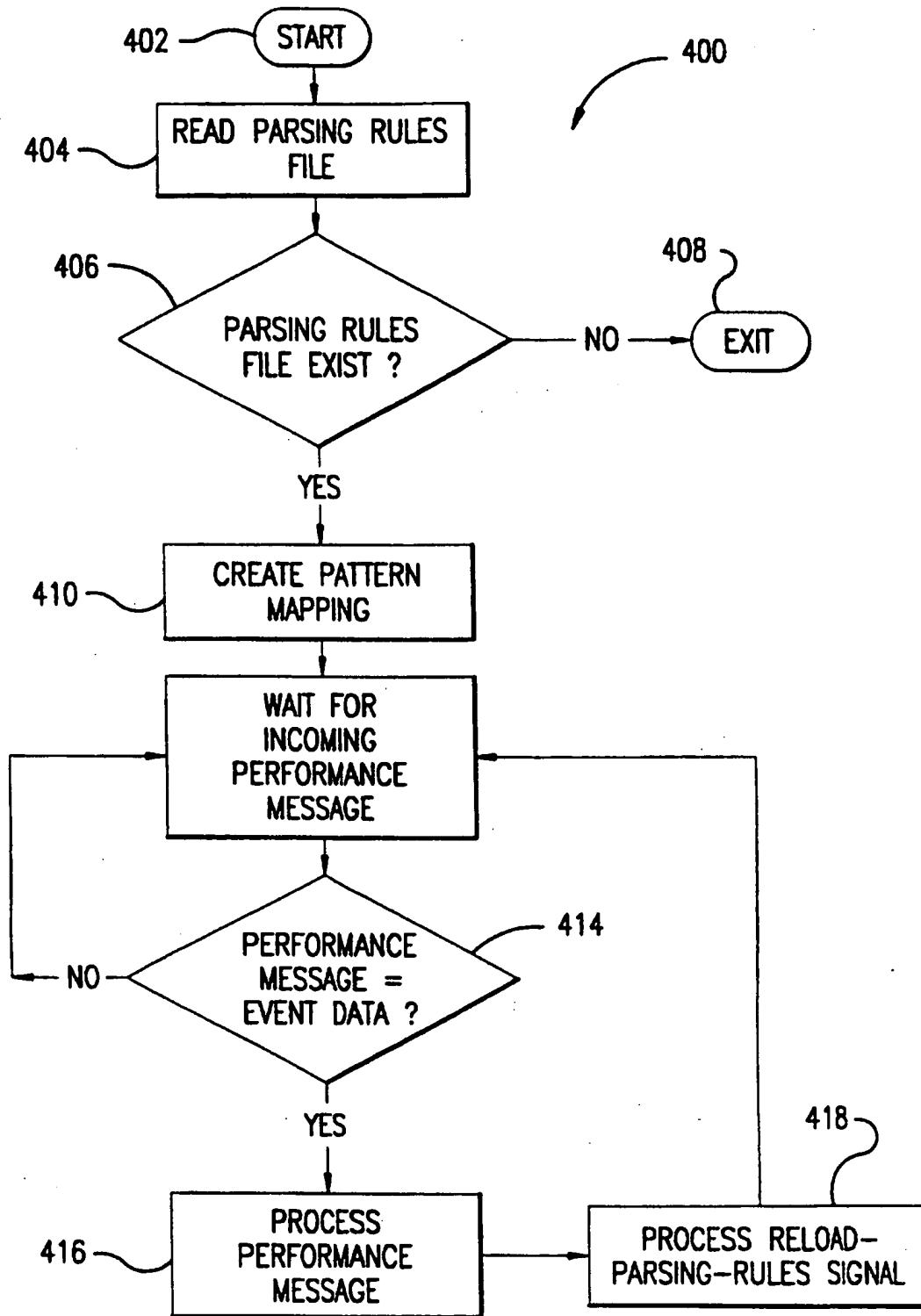


FIG. 4

5/14

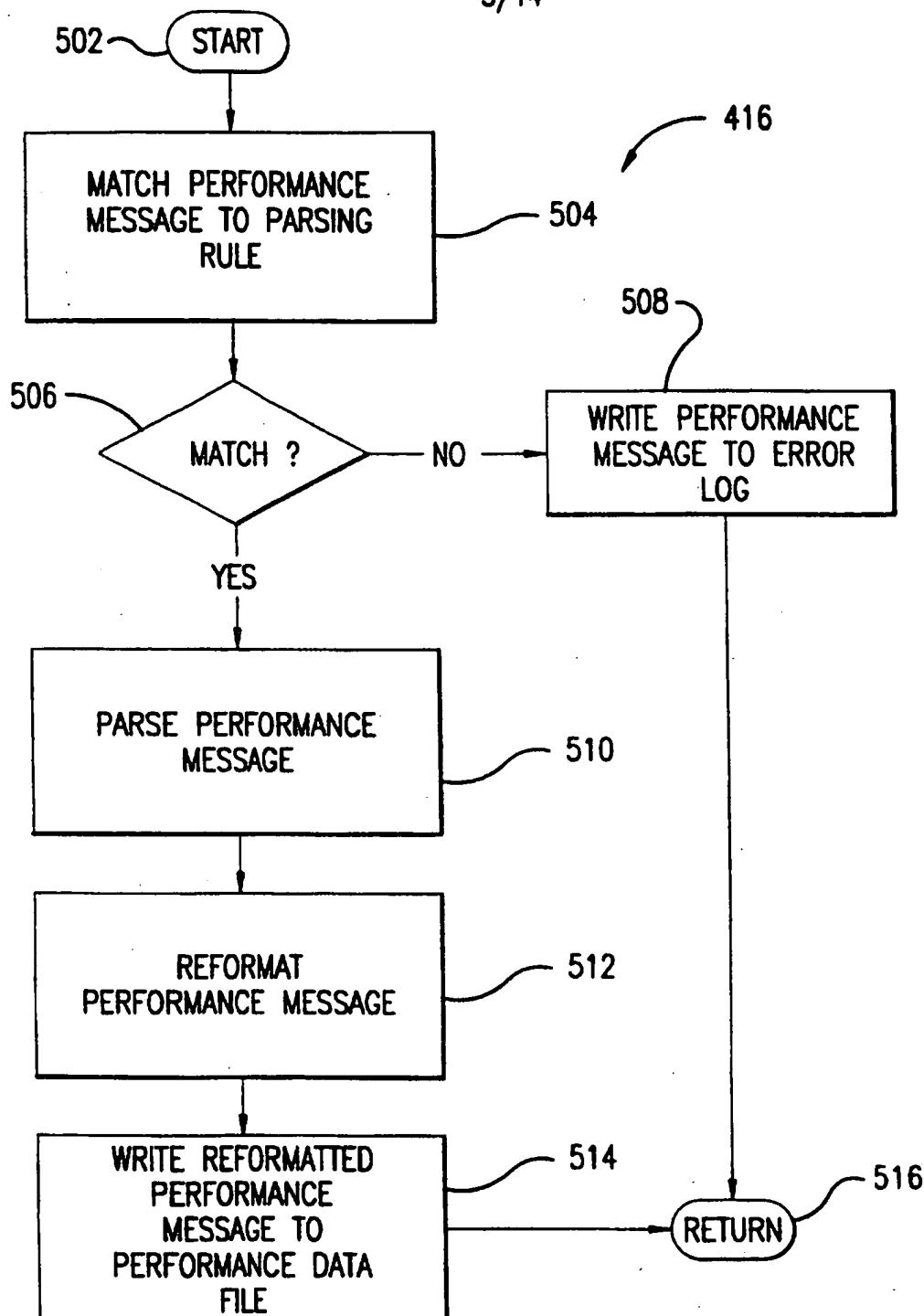
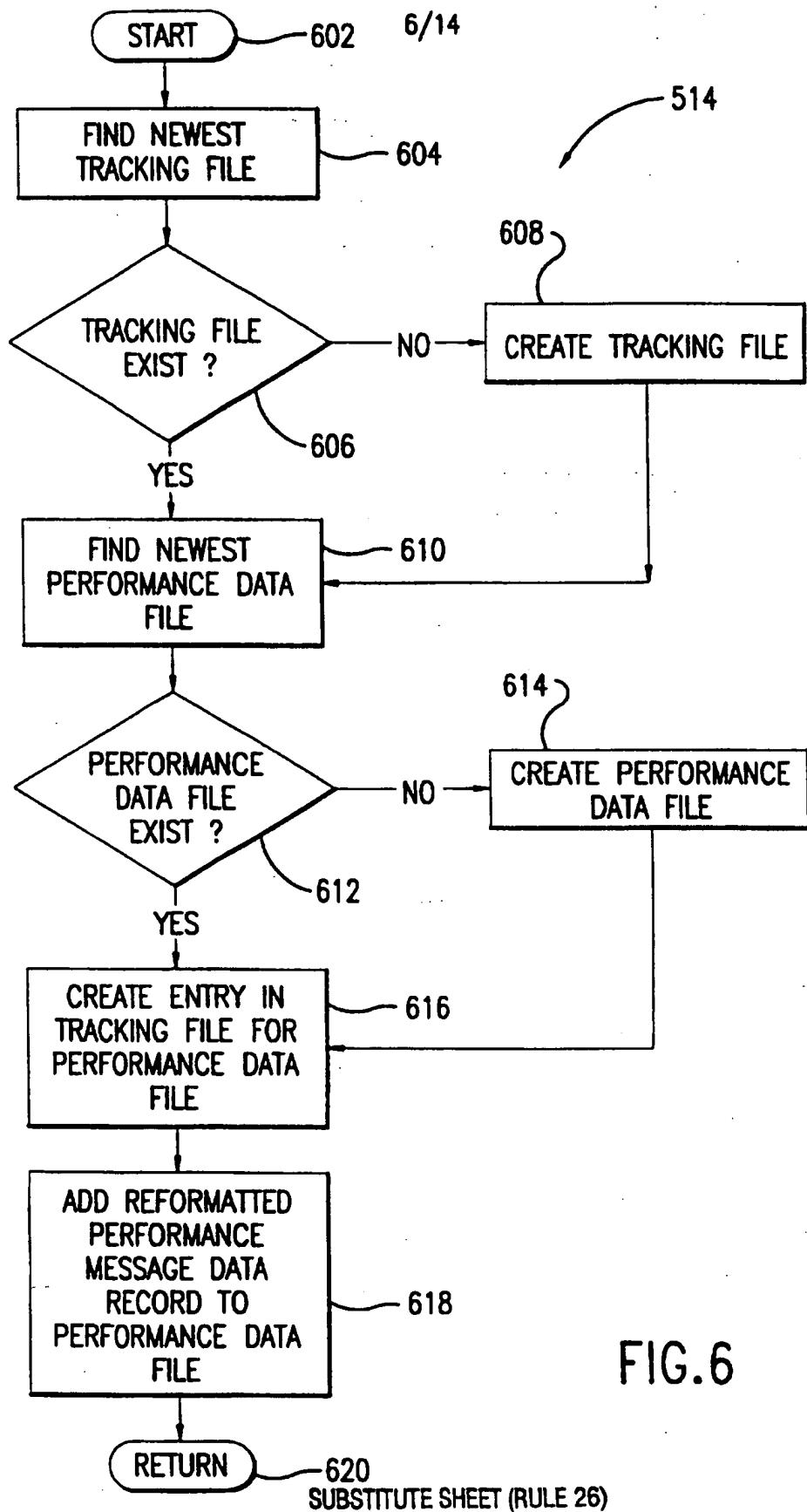


FIG. 5

SUBSTITUTE SHEET (RULE 26)



7/14

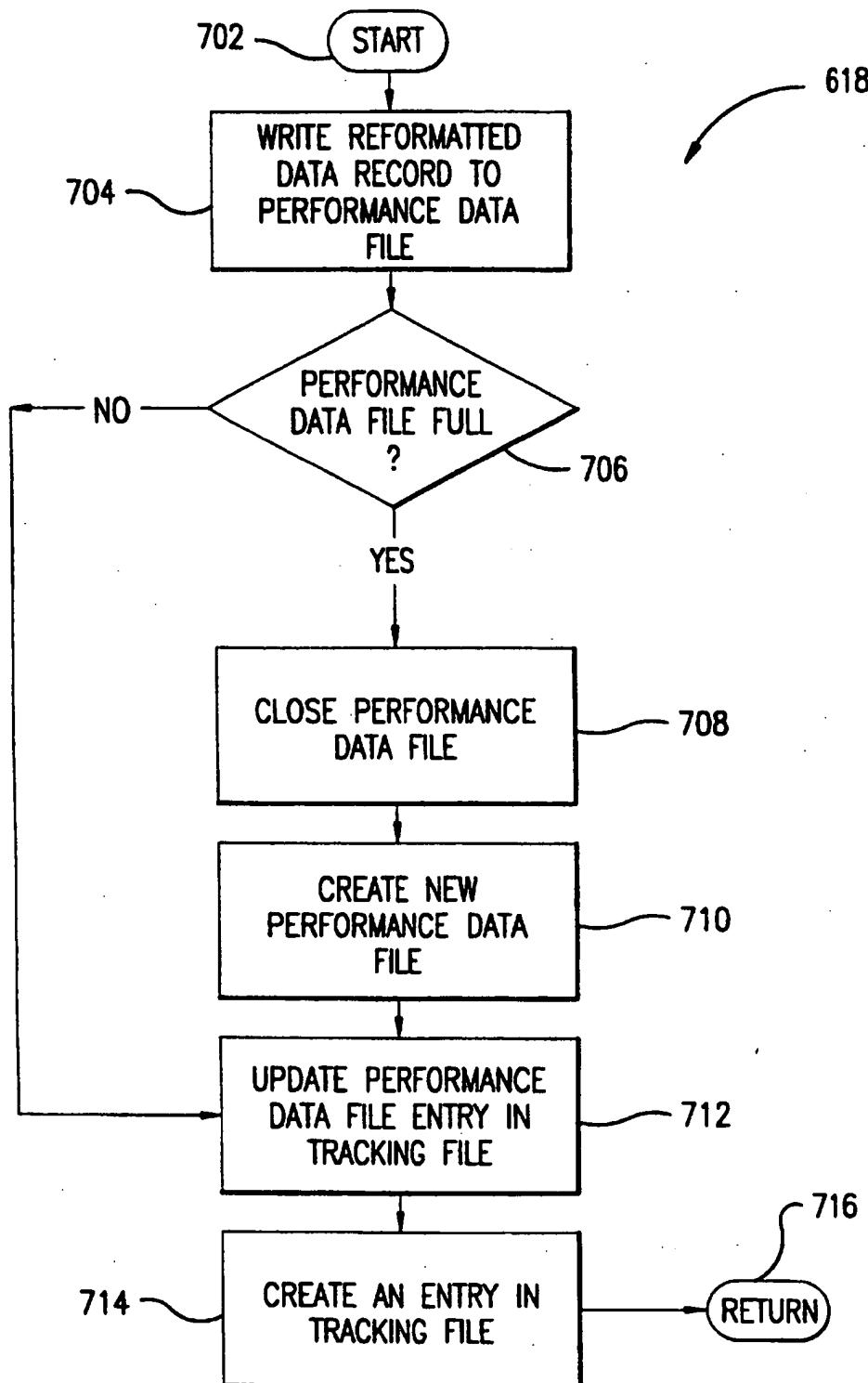


FIG. 7
SUBSTITUTE SHEET (RULE 26)

8/14

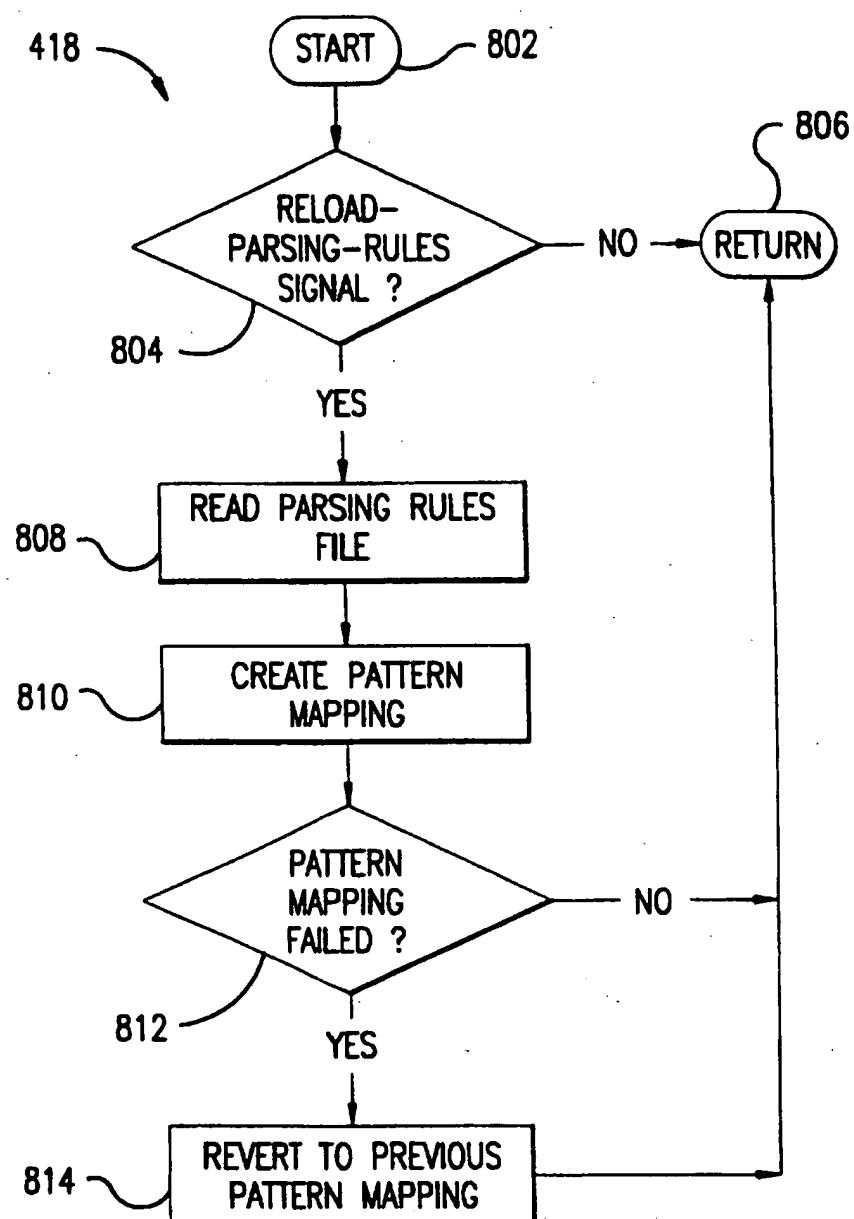


FIG. 8

9/14

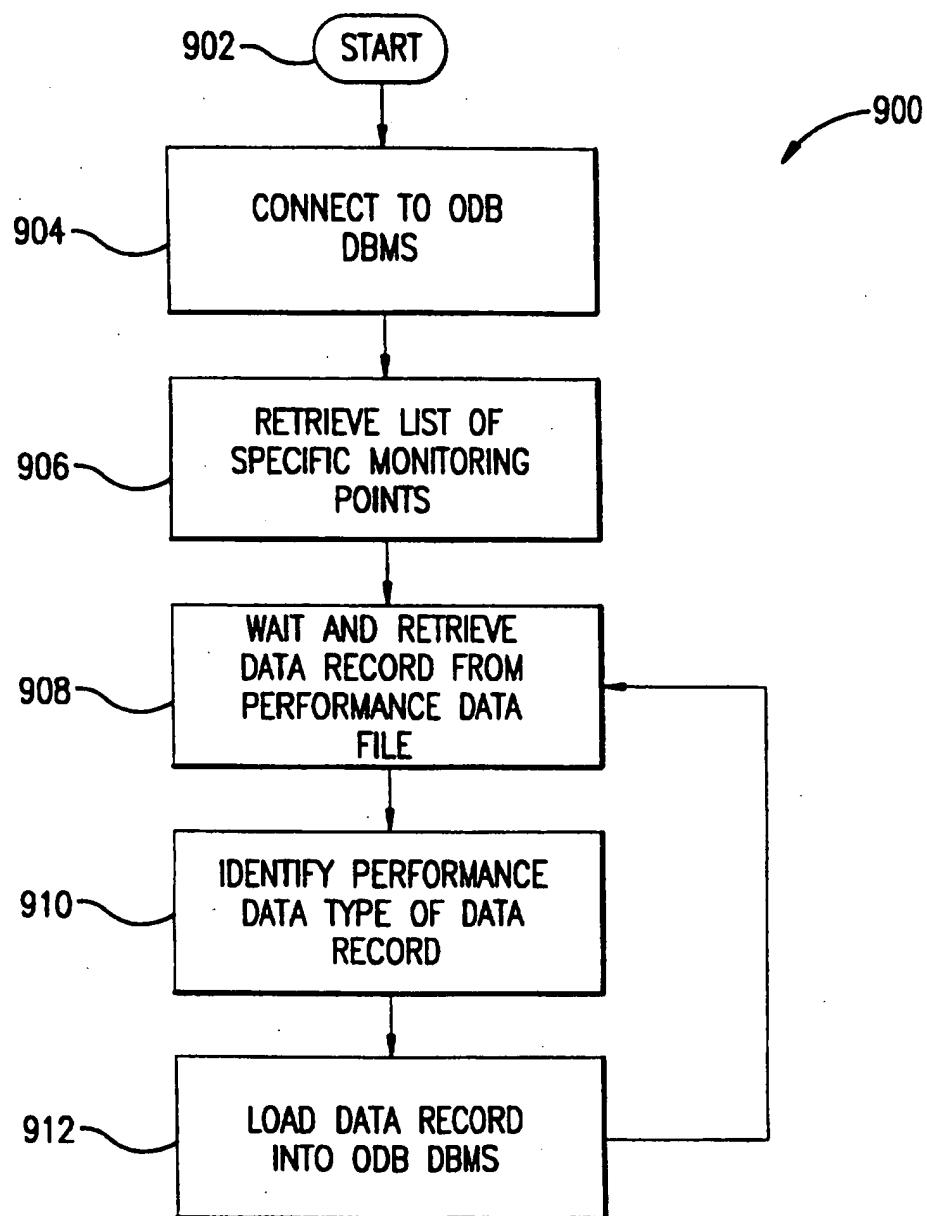


FIG. 9

10/14

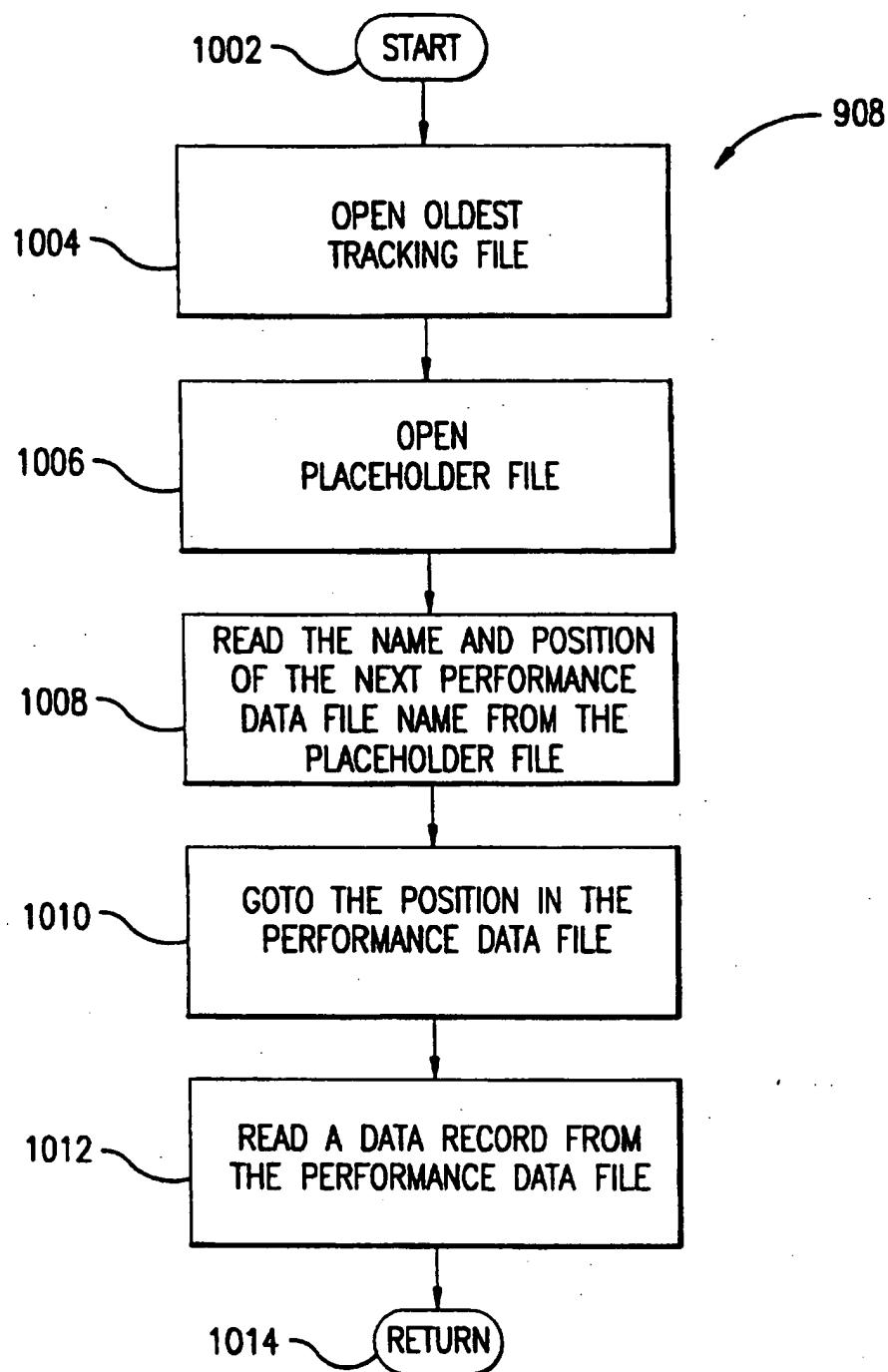


FIG. 10

SUBSTITUTE SHEET (RULE 26)

11/14

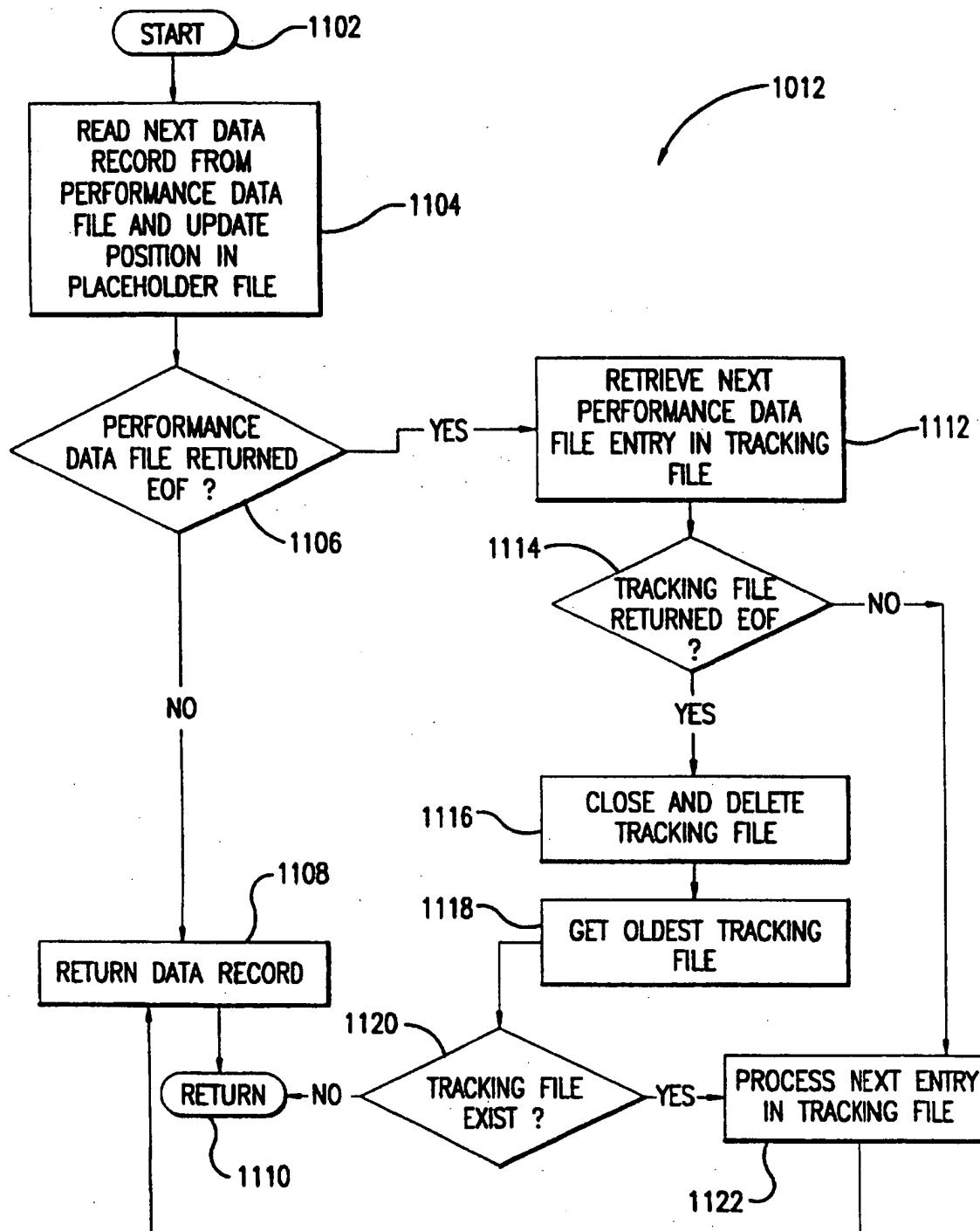


FIG.11
SUBSTITUTE SHEET (RULE 26)

12/14

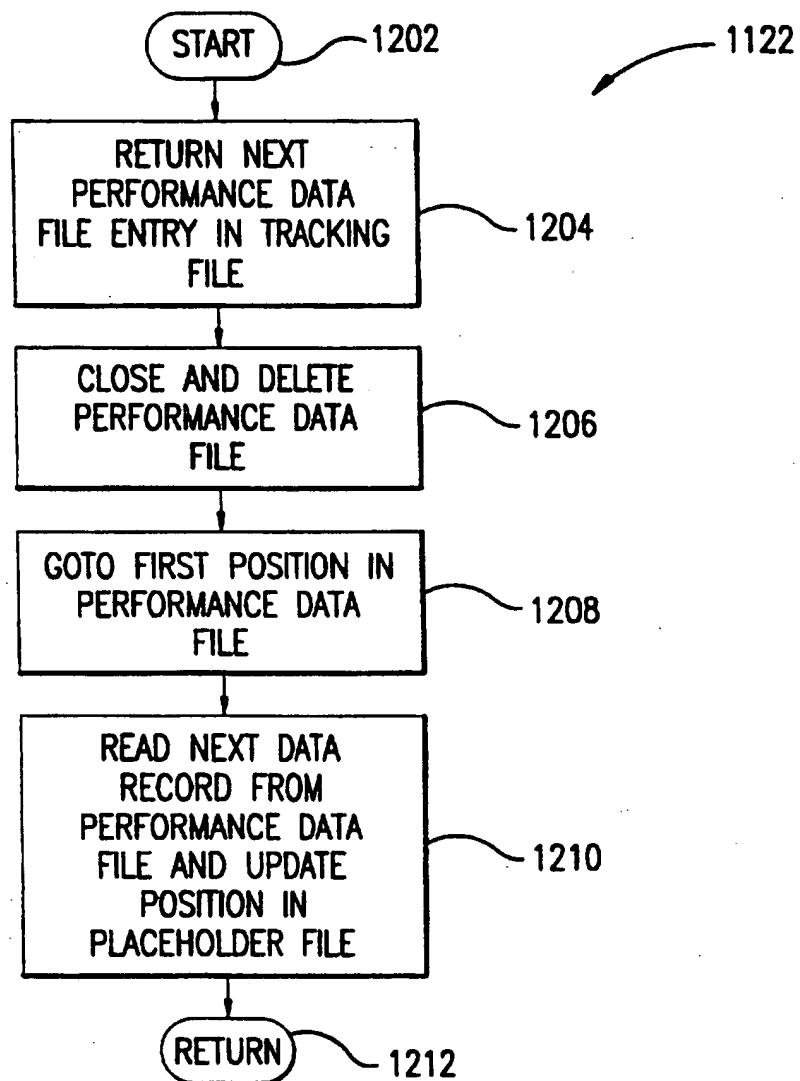


FIG. 12

13/14

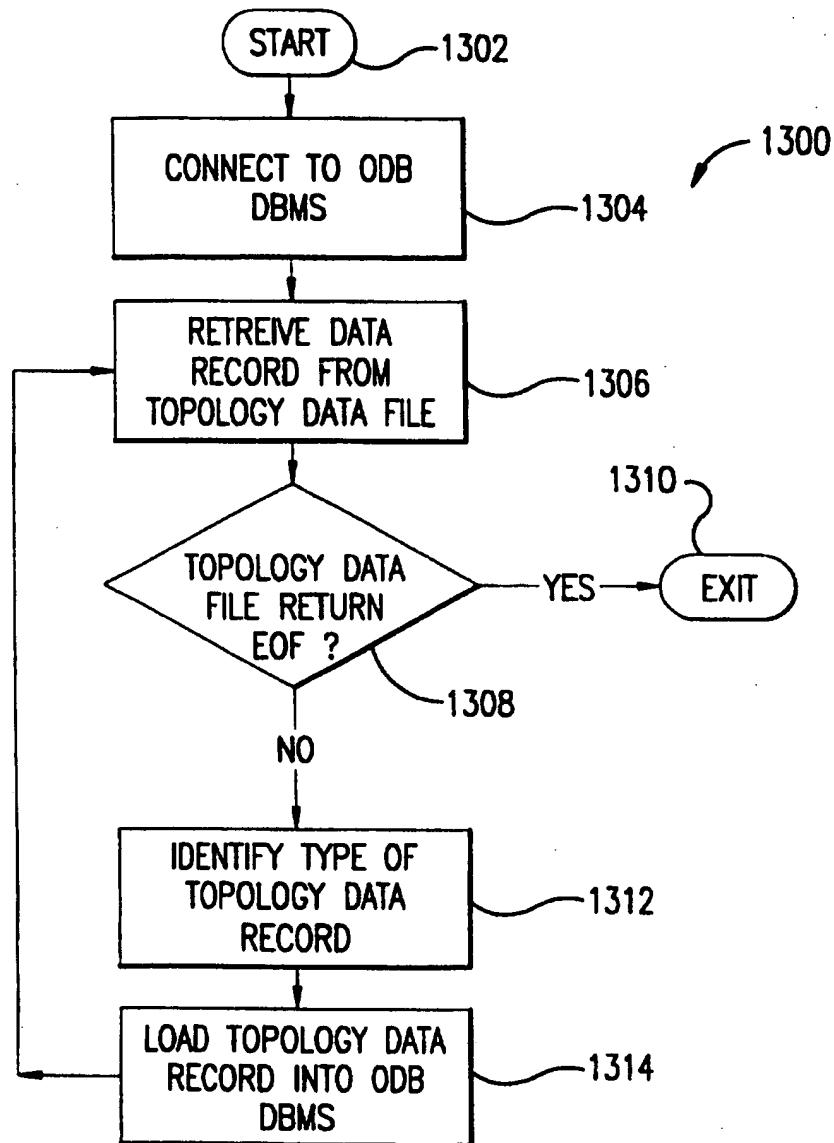


FIG. 13

14/14

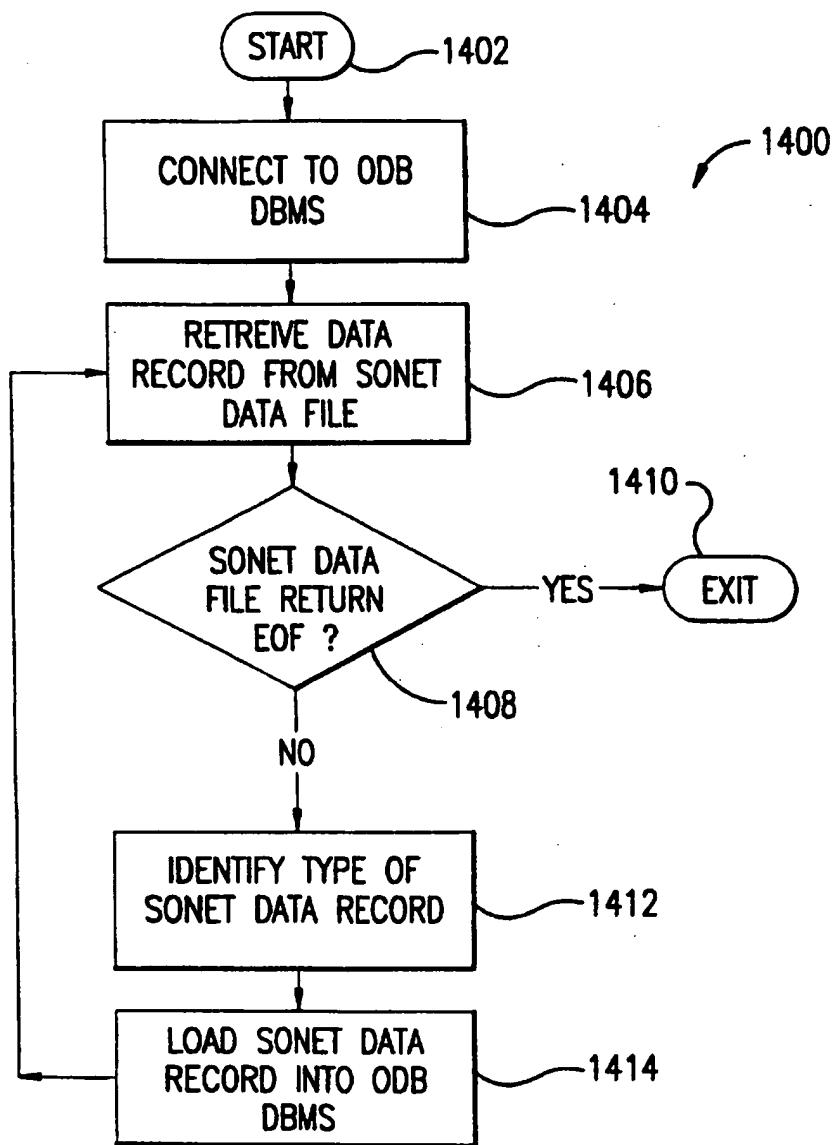


FIG. 14

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/09454

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) : G06F 17/30; H04J 1/16; H04M 1/24, 7/06
US CL : 395/610, 602, 916; 455/9; 364/265, 921.8

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/610, 602, 916, 618, 601, 182.13, 180; 455/9; 364/265, 921.8, 266, 944.9

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|-----------|--|-----------------------|
| Y | US, A, 5,287,505 (CALVERT et al.) 15 February 1994, abstract, column 1 lines 1-25 and 56-68, column 2 lines 1-64, column 6 lines 48-66, figure 2, also note claim 1 | 1-20 |
| Y | US, A, 4,849,879 (CHINNASWAMY et al.) 18 July 1989, abstract, column 1 lines 1-20, column 3 lines 21-62, column 4 lines 50-68, column 5 lines 1-23 and 45-64, figure 2, figures 6 and 7, claim 1 | 1-20 |
| Y | US, A, 5,446,680 (SEKIYA et al.) 29 August 1995, see whole document | 1-20 |

Further documents are listed in the continuation of Box C.

See patent family annex.

| | | |
|-----|--|--|
| • | Special categories of cited documents: | |
| "A" | document defining the general state of the art which is not considered to be part of particular relevance | "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention |
| "E" | earlier document published on or after the international filing date | "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone |
| "L" | document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reasons (as specified) | "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art |
| "O" | document referring to an oral disclosure, use, exhibition or other means | "Z" document member of the same patent family |
| "P" | document published prior to the international filing date but later than the priority date claimed | |

Date of the actual completion of the international search

14 AUGUST 1997

Date of mailing of the international search report

15 SEP 1997

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

Thomas G. Black

Telephone No. (703) 305-9707